CMP
United Business Media

PRODUCT REVIEWS: CODEWARRIOR WIRELESS, WACOM CINTIQ, MORE

OCTOBER 2002

Game Developer

ON THE FRONT LINE OF GAME INNOVATION



Now Hear This!

High-Flying Adaptive Audio

Calling The Shots
Real-Time Cinematography

Postmortem: Z-Axis's
AGGRESSIVE INLINE

Toward Better Scripting

WW
$5.9
0 71486 02133 9
Display Until October 25, 2002
1830
**********AUTO** 3-DIGIT 949
943357012 GD 007 100 0109
NOAH FALSTEIN
THE INSPIRACY
391 N ALMENAR DR
GREENBRAE CA 94904-1153

RULES MAY GOVERN THE WORLD OF TECHNOLOGY BUT IMAGINATION WILL ALWAYS BE KING.



brew™

Customize. Personalize. Realize.™

Put your imagination in motion. The revolutionary wireless platform BREW enables developers – even those with no wireless experience – to write applications and get them to market. No finding a handset partner. No gaining access to chip system software. BREW does it all. Download the BREW SDK at http://brew.qualcomm.com/ZGD1. Wireless carriers need apps. Start thinking.

©2002 QUALCOMM Incorporated. All rights reserved. QUALCOMM is a registered trademark of QUALCOMM Incorporated. BREW and Customize. Personalize. Realize. are trademarks of QUALCOMM Incorporated.

CONTENTS

VOLUME 9, NUMBER 10

October 2002

FEATURES

28 Building an Adaptive Audio Experience

Adaptive audio can enhance the game experience and even foster emergent gameplay. Here Brandon examines how different audio design mechanisms work within various genres, so you can draw your own roadmap toward a higher-quality, more seamless audio experience.

Alexander Brandon

34 Creating an Event-Driven Cinematic Camera: Part 1

Hollywood has spent a century perfecting cinematography techniques. And while developers can use this knowledge in games, they have to rebuild the set 30 times every second. In the first of a two-part series, Hawkins lays the groundwork for an event-driven camera system to highlight action effectively, without becoming a distracting part of it.

Brian Hawkins

42 Postmortem: Z-Axis's Aggressive Inline

Let's see, you want us to create a highly detailed, fast-moving inline skating game featuring almost limitless interactive surfaces and environments. No problem. And, wouldn't it be great to have players design and play in their own levels? Sure. Oh, and can we do this all in one year? (Gulp.) You betcha.

Randy Condon



DEPARTMENTS

2 **GAME PLAN** *Jennifer Olsen*
Turn It Up

4 **INDUSTRY WATCH** *Daniel Huebner*
Barking Dog becomes a Rockstar, Sony takes Incog, Midway sees red, more

6 **PRODUCT REVIEWS**
Metrowerks' CodeWarrior Wireless Studio 7, PMG's Messiah: Animate 3.0, Wacom's Cintiq 18SX tablet

12 **PROFILES** *Alexander Brandon*
Shooting From the Hip: An Interview with Hip Tanaka

COLUMNS

14 **THE INNER PRODUCT** *Jonathan Blow*
Toward Better Scripting

18 **ARTIST'S VIEW** *Hayden Duvall*
Character Matters, Part 2: Getting Ahead of the Curve

24 **SOUND PRINCIPLES** *Jeremy Soule*
What Is Your Game Music Worth?

26 **BETTER BY DESIGN** *Noah Falstein*
Begin at the Middle

56 **SOAPBOX** *Matthew Stibbe*
Games for the People, by the People



COVER *illustration done with Photoshop and pencil drawing (and subsequent scanning into Photoshop). Exclusive to* Game Developer *magazine from Z-Axis.*

Game Developer (ISSN 1073-922X) is published monthly by CMP Media LLC, 600 Harrison St., San Francisco, CA 94107, (415) 947-6000. Please direct advertising and editorial inquiries to this address. Canadian GST: 124513185. **Subscription Rates:** Subscription rate for the U.S. is $49.95 for twelve issues. Countries outside the U.S. must be prepaid in U.S. funds drawn on a U.S. bank or via credit card. Canada/Mexico: $69.95; all other countries: $99.95 (issues shipped via air delivery). Periodical postage paid at San Francisco, CA and additional mailing offices. **Postmaster:** Send address changes to Game Developer, P.O. Box 1274, Skokie, IL 60076-8274. **Customer Service:** For subscription orders and changes of address, call toll-free in the U.S. (800) 250-2429 or fax (847) 647-5972. All other countries call (1) (847) 647-5928 or fax (1) (847) 647-5972. Send payments to *Game Developer*, P.O. Box 1274, Skokie, IL 60076-8274. For back issues write to *Game Developer*, 1601 W. 23rd St. Ste. 200, Lawrence, KS 66046-2703. Call toll-free in the U.S./Canada (800) 444-4881 or fax (785) 841-2624. All other countries call (1) (785) 841-1631 or fax (1) (785) 841-2624. Please remember to indicate *Game Developer* on any correspondence.



GAME PLAN

LETTER FROM THE EDITOR

Turn It Up

October means different things do different people — the leaves change color, autumn breathes a chilly snap into the air, and a million hours are spent sewing the flowing robes of a million Gandalf Halloween costumes. Here at *Game Developer*, October means our annual audio issue.

The past year has seen a number of interesting developments on the game audio front. The proliferation of console development has taken the focus of game audio away from those reviled, tinny speakers sitting on the desktop and into players' increasingly sophisticated home-theater setups in their living rooms. Players so far have liked what they've heard coming through their new consoles, and audio is fast becoming a distinguishing factor among games as graphics gradually declines in dominance on the feature list.

For their part, audio software and hardware companies are continuing to recognize the growing importance of their game audio customers. Other vendors such, as Analog Devices and Sensaura, have admirably attacked some gaping holes in game-audio-specific development tools.

In March, a group of game audio veterans launched the Game Audio Network Guild. Its goal is to foster more and better communication both within the game audio community as well as with other disciplines of game development. Clearly there is a need for some kind of common language between designers, producers, and audio people, as "Can you make that sound bigger?" just doesn't cut it anymore. Also recognizing the need to demystify audio creation among all game developers, *Game Developer* has been running our monthly "Sound Principles" audio column since March.

Few people would choose to watch a movie in total silence or while listening to their favorite CD instead, but most games still give players the option to turn off sound effects and music. But developers should never give players a reason to turn it off, which players will do quickly when a game's audio becomes intolerably repetitive or predictable.

To combat many of the common complaints about game audio, in this month's cover feature, "Building an Adaptive Audio Experience" (p. 28), Ion Storm audio director Alexander Brandon examines several breakthrough games that used different adaptive audio techniques to link game music and sound to story and gameplay in rich, dynamic ways. As much as game audio development can feel like reinventing the wheel each time, there are many valuable lessons existing already about what works and what doesn't.

On the subject of game audio, Alex also did this month's Profile (p. 12), in which he was delighted to interview one of his game-music heroes, Hirokazu "Hip" Tanaka. Hip counts among his influences The Monkees, Burt Bacharach, and reggae dub music, which you can think about next time you're listening to the music and sound effects he did for some of the classic NES/Famicom-era games while he was at Nintendo, such as METROID and KID ICARUS. In addition to apparently having some random Boston-area band named after him, Hip now runs Creatures Inc., where he keeps in touch with Nintendo through work on the Pokemon franchise.

Rounding out the audio focus this month, well-known game composer Jeremy Soule (NEVERWINTER NIGHTS and MORROWIND are among his recent credits) takes a hard look at the business side of game music in our "Sound Principles" column (p. 24), where he asks, "What Is Your Game Music Worth?" If we're going to start suffering from the growing pains and creative tensions of the film and music business, we might as well learn how to make money like they do.

Jennifer

Jennifer Olsen
Editor-In-Chief

GameDeveloper

600 Harrison Street, San Francisco, CA 94107 t: 415.947.6000 f: 415.947.6090

Publisher
Jennifer Pahlka *jpahlka@cmp.com*

EDITORIAL

Editor-In-Chief
Jennifer Olsen *jolsen@cmp.com*

Managing Editor
Everard Strong *estrong@cmp.com*

Production Editor
Olga Zundel *ozundel@cmp.com*

Product Review Editor
Daniel Huebner *dan@gamasutra.com*

Art Director
Elizabeth von Büdingen *evonbudingen@cmp.com*

Editor-At-Large
Chris Hecker *checker@d6.com*

Contributing Editors
Jonathan Blow *jon@bolt-action.com*
Hayden Duvall *hayden@confounding-factor.com*
Noah Falstein *noah@theinspiracy.com*

Advisory Board
Hal Barwood LucasArts
Ellen Guon Beeman Beemania
Andy Gavin Naughty Dog
Joby Otero Luxoflux
Dave Pottinger Ensemble Studios
George Sanger Big Fat Inc.
Harvey Smith Ion Storm
Paul Steed WildTangent

ADVERTISING SALES

Director of Sales/Associate Publisher
Michele Sweeney e: *msweeney@cmp.com* t: 415.947.6217

Senior Account Manager, Eastern Region & Europe
Afton Thatcher e: *athatcher@cmp.com* t: 415.947.6224

Account Manager, Northern California & Southeast
Susan Kirby e: *skirby@cmp.com* t: 415.947.6226

Account Manager, Recruitment
Raelene Maiben e: *rmaiben@cmp.com* t: 415.947.6225

Account Manager, Western Region & Asia
Craig Perreault e: *cperreault@cmp.com* t: 415.947.6223

Account Representative
Aaron Murawski e: *amurawski@cmp.com* t: 415.947.6227

ADVERTISING PRODUCTION

Vice President, Manufacturing Bill Amstutz
Advertising Production Coordinator Kevin Chanel
Reprints Cindy Zauss t: 909.698.1780

GAMA NETWORK MARKETING

Director of Marketing Greg Kerwin
Senior MarCom Manager Jennifer McLean
Marketing Coordinator Scott Lyon

CIRCULATION

BPA INTERNATIONAL
Game Developer is BPA approved

Group Circulation Director Catherine Flynn
Circulation Manager Ron Escobar
Circulation Assistant Ian Hay
Newsstand Analyst Pam Santoro

SUBSCRIPTION SERVICES

For information, order questions, and address changes
t: 800.250.2429 or 847.647.5928 f: 847.647.5972
e: *gamedeveloper@halldata.com*

INTERNATIONAL LICENSING INFORMATION

Mario Salinas
t: 650.513.4234 f: 650.513.4482 e: *msalinas@cmp.com*

CMP MEDIA MANAGEMENT

President & CEO Gary Marshall
Executive Vice President & CFO John Day
Chief Operating Officer Steve Weitzner
Chief Information Officer Mike Mikos
President, Technology Solutions Group Robert Faletra
President, Business Technology Group Adam K. Marder
President, Healthcare Group Vicki Masseria
President, Electronics Group Jeff Patterson
President, Specialized Technologies Group Regina Starr Ridley
Senior Vice President, Global Sales & Marketing Bill Howard
Senior Vice President, HR & Communications Leah Landro
Vice President & General Counsel Sandra Grayson
Vice President, Creative Technologies Philip Chapnick

CMP United Business Media

GamaNetwork

WWW.GAMANETWORK.COM

"[REAL-TIME SHADERS] ALLOW FOR PREVIEWING OF EFFECTS ON THE XSI VIEWPORT **DURING CONTENT CREATION**, SOMETHING THAT HAS PREVIOUSLY ONLY BEEN POSSIBLE ON **THE XBOX**."
– Yuichi Masuya, Microsoft Co., Ltd., Xbox Division



JSRF – Jet Set Radio Future Original Game ©SEGA ©SEGA/Smilebit 2002

WHAT YOU SEE IS WHAT YOU GET. ONLY IN **XSI**™ v.**2.0**

Artists and developers, here's the game plan: Start with **SOFTIMAGE|XSI** v.**2.0**, the only true creative choice for sophisticated real-time effects.

XSI Real-Time Shaders deliver **state-of-the-art shading** control along with the industry's only native **nonlinear animation (NLA)** system for action-based motion. Now you can work with your art the same way your game engine does.

Feel like calling the shots yourself? XSI v.2.0 internals are fully exposed with the new integrated **Software Development Kit (SDK)** and file I/O is seamless via the free, extensible **dotXSI**™ **file format**.

Ready to play? Visit **www.softimage.com/v2**, order the new XSI Experience CD and **let the games begin**.

2.0

SOFTIMAGE® | XSI™

Innovate. Create. Collaborate.

AMD Athlon MP
AMD Certified

© 2002 Avid Technology, Inc. All rights reserved. Softimage and Avid are registered trademarks and XSI, the XSI Logo, dotXSI and Make, Manage and Move Media are trademarks of Avid Technology, Inc. AMD, the AMD logo, AMD Athlon MP and combinations thereof are trademarks of Advanced Micro Devices, Inc. All other trademarks contained herein are the property of their respective owners.

Avid
Make, Manage and Move Media™

INDUSTRY WATCH

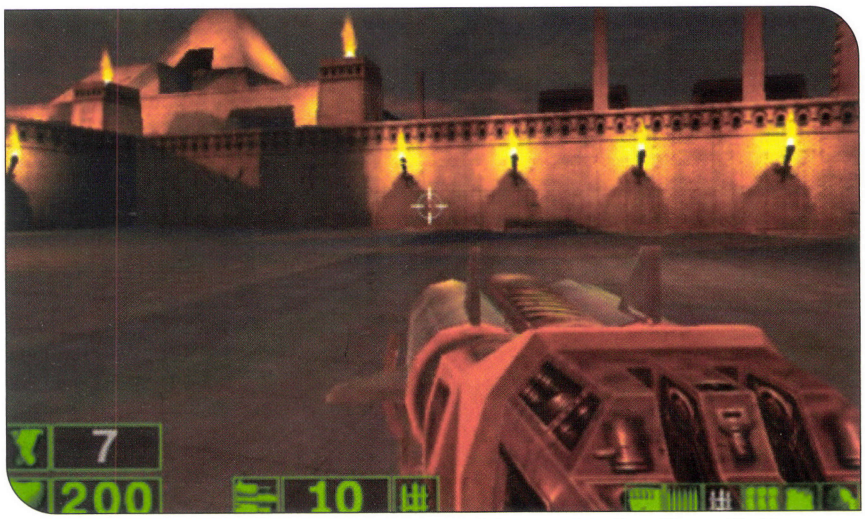
THE BUZZ ABOUT THE GAME BIZ | daniel huebner

Take Two buys Barking Dog, Sony picks up Incog. Take-Two, the corporate parent of Rockstar Games, acquired Vancouver-based Barking Dog for $3 million in cash and 242,450 shares of restricted common stock. Barking Dog, currently working on a title for Rockstar, employs around 50 developers in three different teams. Barking Dog will take on the name Rockstar Vancouver, bringing the number of Rockstar studios to three. The studio located in Oakville, Ontario (formerly known as Rockstar Canada), will now be called Rockstar Toronto.

Sony Computer Entertainment America has acquired TWISTED METAL developer Incog. Salt Lake City–based Incog Inc., formerly known as Incognito Studios, will become a Sony internal studio with all current employees staying on in Incog's Utah location and under the Incog name. Incog's TWISTED METAL franchise has sold more than 5 million units in its lifetime, making the developer a tempting target in Sony's bid to strengthen its internal development efforts. The terms of the deal were not disclosed.

In the black. Electronic Arts announced its first-quarter results, posting consolidated net revenues of $331.9 million, an increase of 82 percent from last year. Net income for the quarter rebounded to $7.4 million from a net loss of $45.3 million last year. The company cited higher sales for PC and PS2 games as a major reason for its strong performance during the quarter. EA.com, however, continues to lose money. Net revenues at EA.com were up 21 percent to $19.8 million, but EA's online division still tallied a $12.8 million loss.

Activision also posted strong first-quarter results, with net revenues up 73 percent to $191.3 million. Net income for the quarter reached $20.7 million. In addition to the sterling financial results, Activision's first quarter also included the acquisition of DAVE MIRRA BMX developer Z-Axis, the purchase of a 30-percent stake in developer Infinity Ward, the securing of distribution rights to id's DOOM III, and a contract extension with Tony Hawk until 2015. Based on the favorable first quarter, Activision raised its fiscal-year revenue projections from $890 million to $920 million.



SERIOUS SAM TAKES TWO. The SERIOUS SAM series will be one of the first titles tackled by Gotham Games, Take-Two's newest label.

THQ joined EA and Activision in posting unexpectedly strong results. The company reported second-quarter net income of $4.9 million, an increase of 43 percent from last year. Revenue hit $85.8 million, up 55 percent from last year. These results exclude a previously announced non-cash charge of $2.6 million (7 cents per share), for the discontinuation of the Network Interactive Sports (NIS) online joint venture in the United Kingdom. THQ also increased its 2002 guidance for revenue from $515 million to $525 million.

In the red. Midway posted strongly improved second-quarter revenues but was still unable to stop the red ink. The company's revenues hit $28.1 million, a healthy improvement from the same period one year ago, but it wasn't enough to keep Midway from posting a pretax loss of $10.7 million. With consolidation and stock charges tallied, Midway's net loss reached $27.9 million. The company shipped nine titles in the second quarter and anticipates shipping 10 more in the coming quarter, leading the company to maintain its fiscal-year projection of a pretax income of $25 to $33 million.

Take-Two adds value. In addition to beefing up its Rockstar label, Take-Two is making moves to secure its place in the value segment. The company is launching a new label, Gotham Games, to focus on mass-market and value-priced titles. The first titles under the Gotham imprint include SERIOUS SAM for Xbox and CELEBRITY DEATHMATCH for PSX.

UPCOMING EVENTS CALENDAR

INTERACTIVE ENTERTAINMENT FORUM

SCANDIC HOTEL
Copenhagen, Denmark
October 14–15, 2002
Cost: €2,518 plus tax
www.marcusevansuk.com

PROJECT BAR-B-Q

GUADALUPE RIVER RANCH
Boerne, Tex.
October 17–15, 2002
Cost: $1,850–$2,200
www.projectbarbq.com

Hero maker.



Your creative awards called. Said they're waiting to happen on the Evo Workstation W8000, from the new HP. Designed and optimized for intensive graphics and the latest video applications, this HP Evo Workstation with an Intel® Xeon™ Processor turns creative juices into rivers. This, plus our legendary software partnerships, now means that designers, animators, editors and developers have even more reason to get all giddy. Mix with way-out-there ideas and clear some shelf space for those awards.

Ready to strut your stuff?
Visit www.hp.com/go/evoworkstations
or call 1-800-888-0414

intel inside XEON

The Power of
MACROPROCESSING

hp® invent

HP PCs use genuine Microsoft® Windows®
www.microsoft.com/piracy/howtotell

©2002 Hewlett-Packard Company. During the HP/Compaq product transitions, some HP workstation products and packaging may be labeled with the Compaq brand. Intel, the Intel Inside logo and Xeon are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries. Screen image courtesy of Softimage Co. and Avid Technology Inc. Softimage/XSI is a trademark or registered trademark of Softimage Co. and Avid Technology Inc.

Metrowerks' CodeWarrior Wireless Studio 7

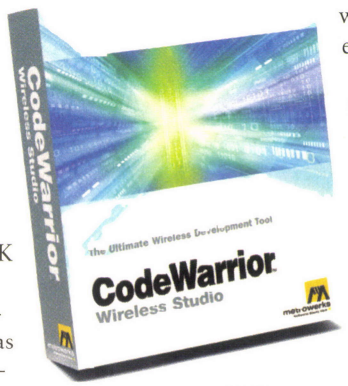
by ralph barbagallo

Why should you spend a dime on wireless Java development tools when Sun provides their own kit for free? Good question. For wireless game developers, Metrowerks' CodeWarrior Wireless Studio 7 definitely serves to fill a growing need. Wireless games are getting more complicated. Memory capacities are rising and multimedia capabilities are improving, and the spare-bedroom programmer banging out a mobile game in a few weekends will have to get serious. This necessitates things like debuggers, source control, and having the ability to quickly build your code for multiple handsets.

Slowly, we are rising from the primordial soup of stale WAP products and simplistic embedded games, but even within a leading mobile applications platform standard such as Sun's Java 2 Micro Edition (J2ME), there's still a mess of custom extensions and handset features. Extensions to access features such as MIDI music, sprite graphics, pixel transparency, and low-level networking are commonplace in various J2ME SDKs. CodeWarrior Wireless Studio 7 has taken on the daunting task of uniting all of these disparate SDK extensions.

Opening the Box

CodeWarrior Wireless Studio 7 Professional Edition comes on two discs. The first has the actual IDE on it while the second CD, labeled "SDK Disc," contains the J2ME SDKs and emulators for Sun, Motorola, Siemens, and Sprint PCS. It also includes Sun's PersonalJava SDK as well as support for the Sharp Zaurus platform. The SDK disc has a few third-party libraries, including PointBase Micro Edition, Softwired iBus-Mobile LE, Lutris Enhydra kXML, and kSOAP. This also includes the popular Retroguard obfuscator. Many of these additions are of minor importance to the average game developer. However, Retroguard is invaluable when trying to slim down your application for distribution.



Using Wireless Studio

CodeWarrior Wireless Studio works largely the same way as their normal Java IDE. In fact, you can also create Java 2 Standard Edition (J2SE) projects with it. Think of it as Java CodeWarrior; it includes all of the standard features of editing, project management, and version control integration, and adds mobile development features such as editing JAD properties, verifying and packaging the MIDlet suite, obfuscating, and developing your code with a variety of different emulators and SDKs.

That last feature is the major advantage of Wireless Studio 7. The J2ME world is currently awash in nonstandard extensions and unique hardware implementations. CodeWarrior has consolidated many of these custom SDKs and emulators under one roof. Simply pick which SDK you want to compile with and you are in business. CodeWarrior also will work with updated versions of some SDKs. In particular, newer offerings such as version 2.0 of Motorola's J2ME SDK have installation options to merge automatically with CodeWarrior. Hopefully more manufacturers will follow suit.

CodeWarrior Wireless Studio is, in many cases, at the mercy of the quality control of another company's SDK. Some of the SDKs, such as with Sprint PCS's toolkit, are genuine beta releases, while others are finished products with a lot of quirks. Even though Motorola now owns Metrowerks, support for Motorola hardware is the worst of the lot. Metrowerks wisely allows for switching between any of the provided kits, as well as the ability to register different VMs and SDKs to expand support for other devices.

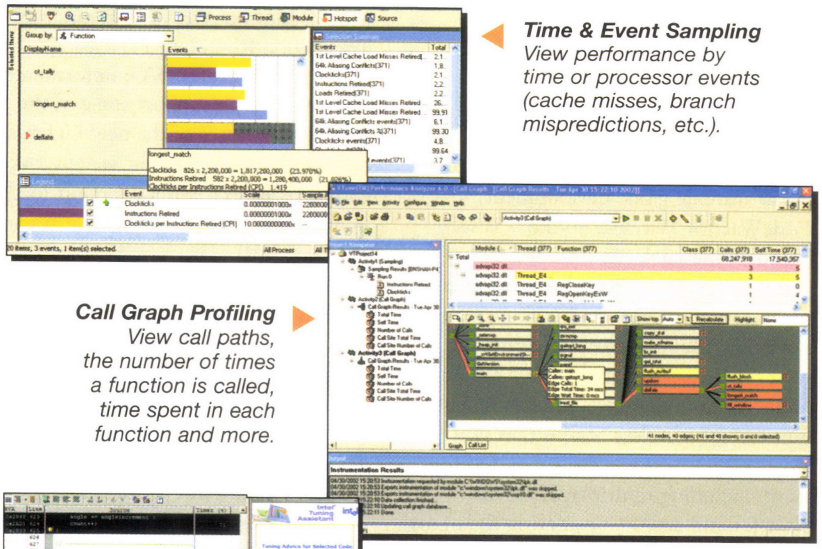
Wireless Studio 7 comes with CodeWarrior's familiar debugging tools. This includes their robust source-level debugger with all the pleasantries we have grown accustomed to in a modern debugger. However, one of the more

RALPH BARBAGALLO | *Formerly of Ion Storm, Ralph runs Flarb Development, a wireless gaming studio in Southern California. He currently is finishing up a book,* Wireless Game Development in C/C++ with BREW *(Wordware), for release later this year.*

Maximize Your Application Performance
with Intel® Software Development Products

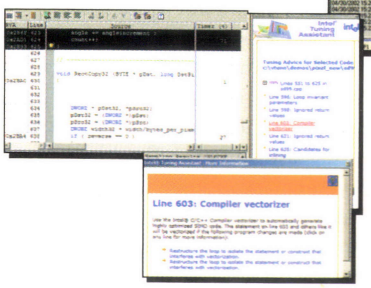
Intel, the world's leading processor vendor, provides world class software development tools. These tools are designed to integrate easily with the development environment you're already using. All tools include 1 year of Intel Premier Support. Take advantage of the advanced features of Intel® Pentium® 4, Intel® Xeon™, and Intel® Itanium® processors with tools from the people who know processors best.





Time & Event Sampling
View performance by time or processor events (cache misses, branch mispredictions, etc.).

Call Graph Profiling
View call paths, the number of times a function is called, time spent in each function and more.



Intel Tuning Assistant
Uses Intel compiler technology to give advice on improving code performance.

Intel® VTune™ Performance Analyzer

New Version 6

VTune™ Performance Analyzer 6 helps you increase your software performance by enabling you to identify and analyze bottlenecks in your code. The VTune analyzer has many powerful analysis features that offer useful guidance on how to improve performance. Version 6 supports VS.NET*, C# and multi-threaded applications.

> ***"The Intel VTune Performance Analyzer took a multi-day task and turned it into a sub-day task."***
>
> **—Randy Camp**
> *V.P. Software Research and Development*
> *MUSICMATCH, Inc.*

Intel® C++ and Fortran Compilers for Windows* and Linux*

The Intel® C++ and Fortran Compilers make it easy to get outstanding performance on Intel® architecture. Intel compilers provide access to advanced processor technologies with no code modifications necessary. Our compilers are also designed to be compatible with leading tools and development environments like GNU C and Microsoft Visual Studio*.

Intel® Performance Libraries

The Intel Performance Libraries simplify development of high-performance applications. These libraries free developers from having to write separate versions of code for different processors by providing a simple, consistent API interface that works across processor architectures. The libraries include highly optimized processor-specific code for graphics, multimedia, mathematical and signal processing functions.

YOU SAVE UP TO $164!	Paradise #	Retail	Discount
Intel® VTune™ Performance Analyzer 6	I23 0170	$699.00	$534.99
Intel® C++ Compiler 6.0 for Windows*	I23 016G	$399.00	$305.99
Intel® Math Kernel Library for Windows*	I23 0180	$199.00	$173.99

* *Also available for Linux*

FREE 30-day trials available at:
programmersparadise.com/intel

Programmer's Paradise®
To order, or request additional information call:
800-423-9990
Email: intel@programmers.com

© 2002 Intel Corporation Intel, the Intel logo, Pentium, Itanium, Intel Xeon and VTune are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries. *Other names and brands may be claimed as the property of others.

02CT10C

★★★★★ excellent
★★★★ very good
★★★ average
★★ disappointing
★ don't bother

interesting features is the ability to do on-device debugging. Emulators are often extremely inaccurate when compared with an actual handset, making it necessary to debug your code extensively on real hardware before releasing a commercial application.

Previously, you have only been able to get `println` output via a terminal program communicating from the handset to a host PC. CodeWarrior Wireless Studio 7 includes support for debugging on the device itself. This means you can connect the debugger to the handset and step through code as it executes on the actual hardware. This requires a debug version of the handset firmware to be installed on the device. Right now, only debug Java VMs for PDAs are supported with on-device debugging. Metrowerks is working with manufacturers to provide developer firmware updates that work with CodeWarrior's debugger. This is an absolute godsend and, if it works, is alone worth the price of admission.

CODEWARRIOR WIRELESS STUDIO 7 ★★★★

STATS

METROWERKS
Austin, TX
(512) 997-4901
www.metrowerks.com

PRICE
$599 (MSRP)

SYSTEM REQUIREMENTS
PC Software: Windows 98/2000/Me/NT 4.0 with Service Pack 4 or later, Sun Microsystems' Java SDK, 1.2 or later (JSDK 1.3.1 and JSDK 1.4.0 included). Intel Pentium class or AMD-K6 class processor, 64MB RAM, 250MB free hard disk space, CD-ROM drive for installation.

PROS

1. Unites many different J2ME SDKs under one environment.
2. Price is very reasonable compared to other similar products.
3. On-device debugging will be the ultimate feature once more manufacturers support it.

CONS

1. Third-party J2ME SDK integrations vary widely in quality.
2. CodeWarrior interface — either you love it or you hate it.
3. On-device debugging only works on one device (at the moment).

Ahead of the Game

When it comes to J2ME IDEs for mobile game development, CodeWarrior Wireless Studio 7 is way ahead of the pack. At a mere $599 (MSRP), it's an absolute steal when compared to competing products, such as Borland's JBuilder, that can approach prices of $2,000. CodeWarrior has announced an upcoming "Entertainment Edition" of Wireless Studio that will have specific features for game development, including multiplayer server libraries and such. Considering the miniscule budgets of the average mobile game, CodeWarrior's low price and useful array of features make it the only choice for cash-strapped development houses of the mobile age.

PMG'S MESSIAH: ANIMATE 3.0

by sergio rosas

With version 3.0, PMG's Messiah is no longer just a character animation plug-in for Lightwave 3D, but is now a full-featured 3D animation package busting at the seams with great features.

The biggest improvement to Messiah is that it is now a stand-alone program, with added support for multiple rendering pipelines. Straight out of the box, Messiah animations can be exported for rendering in Lightwave, 3DS Max, Maya, and Softimage XSI. Don't get too excited, though, because Messiah animation data gets exported to the other rendering packages as vertex deformations only. In other words, you load up your model in the 3D package you want to render in and apply the Messiah plug-in; the plug-in then deforms your model's vertices frame by frame. Still, this is the best effort I have seen among any 3D package to try to play well with others.

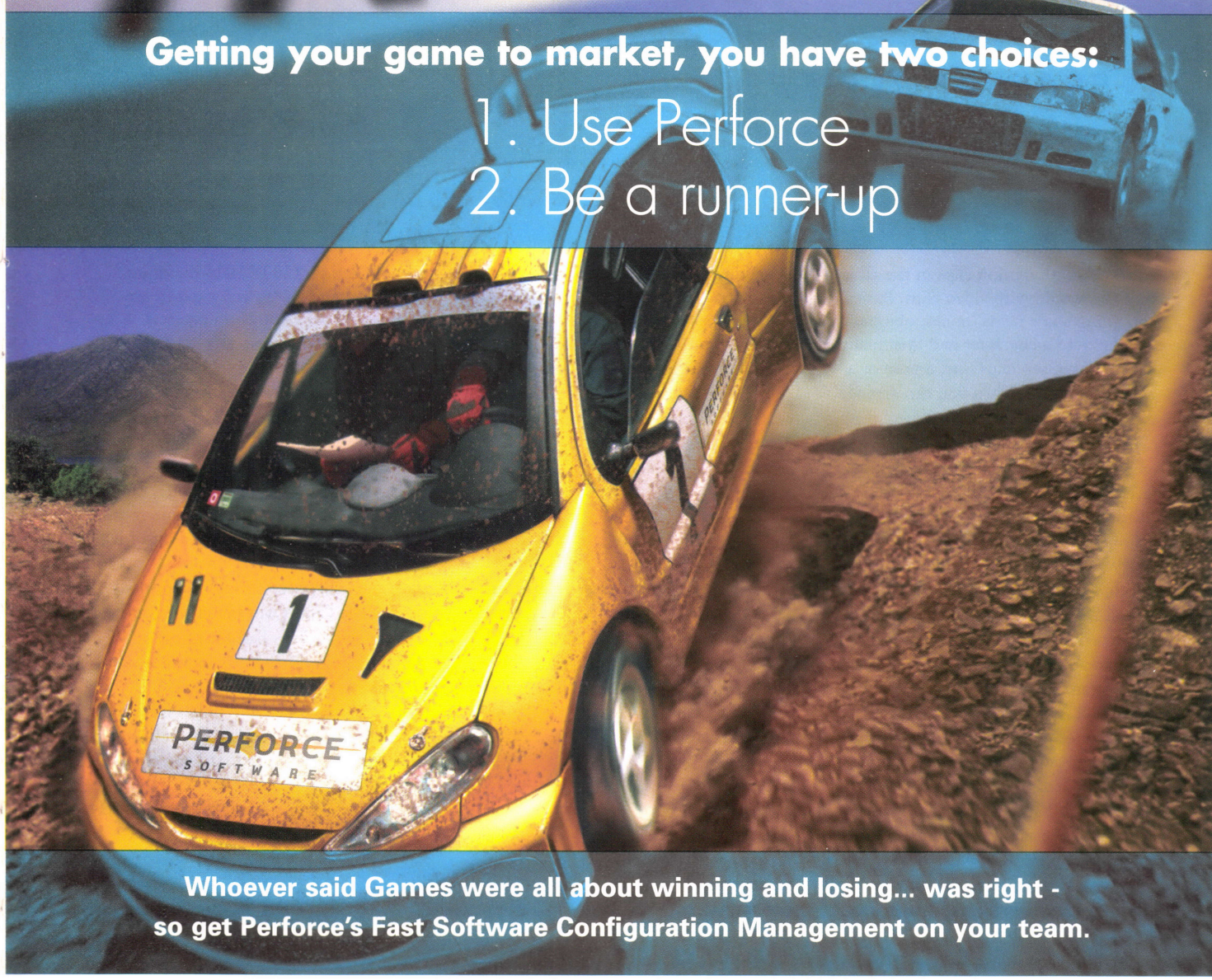
Messiah can import from a variety of formats, including 3DS (.3DS), BioVision Mocap Data (.BVH), DXF objects (.DXF), Messiah Motion (.FXM), Messiah Scene (.FXS), Motion Analysis Hierarchical Translation Rotation (.HTR), Lightwave 5.x or 6.x objects (.LWO; unfortunately, Messiah doesn't support some of the advanced features of the new Lightwave objects such as skelegons, weight maps, or endomorphs, but it does support loading in individual object layers), and Wavefront objects (.OBJ).

Messiah has the ability to export its skeletal weight and animation data to ASCII format, making it very easy for programmers to do what they need with it. Additionally, Messiah has added a full scripting language and an SDK to allow developers to create their plug-ins or pipelines in and out of Messiah.

Messiah's strength is apparent when setting up advanced character rigs. Setting up skeletons is similar to the way you draw bones in the Lightwave layout, except with a more refined interface. The program has a Setup tab that takes the character back to its rest position for easy bone editing. You can actually start animating a character and then go back to the Setup tab to modify a bone or add an extra bone or muscle, all without losing your animation. Messiah's bones are amazingly fast, even with complicated IK and expressions on a rig. Although the bones deformation envelope works well, Messiah doesn't currently have the ability to create or import weight maps.

The program has a very powerful implementation of expressions, shipping with a ton of predefined expression functions that are simple plug-and-play, even for nonmathematicians. IK, sliders, and morph targets are straightforward to implement and can all be tied into expressions as well. One neat feature new to Animate 3.0 is the premade Setups panel, which comes with some setups, and with one double-click, you can also add your own.

Messiah has a built in nonlinear animation (NLA) system called Compose. It lets



Perforce's Software Configuration Management System is the choice of winners, because no other tool can match the speed, control and reliability that it brings to the management of digital assets (binary and text files).

Maintain your top speed no matter how many users or files. At the core of Perforce lies a relational database with well-keyed tables, so simple operations can be accomplished in near-zero time. Larger operations (like labeling a release and branching) are translated into keyed data access, giving Perforce the scalability that big projects require.

Install it fast, learn it fast, execute operations fast. With other SCM systems, developers face an unpleasant choice: do it the right way or do it the fast way. Perforce's speed and reliability mean the fast way is always the right way.

Work anywhere. Perforce is efficient over high-latency networks such as WANs, the Internet and even low-speed dial-up connections. Requiring only TCP/IP, Perforce makes use of a well-tuned streaming message protocol for synchronising client workspace with server repository contents.

Develop and maintain multiple codelines. Perforce Inter-File Branching™ lets you merge new features and apply fixes between codelines. Smart metadata keeps track of your evolving projects even while they develop in parallel.

Supports all development platforms. Perforce runs on more than 50 operating systems, including all flavours of Windows, UNIX, Linux, Mac OS X and more.

Integrate with leading IDEs and defect trackers: Visual Studio.NET®, Visual C++®, Visual Basic®, JBuilder®, CodeWarrior®, TeamTrack®, Bugzilla™, ControlCenter®, DevTrack® packages, and more.

PERFORCE
SOFTWARE

Fast Software Configuration Management www.perforce.com

Download your free 2-user, non-expiring, full-featured copy now from www.perforce.com
Free (and friendly) technical support is on hand to answer any and all evaluation questions.

All trademarks used herein are either the trademarks or registered trademarks of their respective owners.

★★★★★ excellent
★★★★ very good
★★★ average
★★ disappointing
★ don't bother

you work with motion clips in the same way you would with video clips. Basically you group some items (bones, objects, and the like) into a character and then make a library of motion clips (say, a pose or a walk cycle) for it. Then you can arrange the clips in any order and/or blend the clips into one another using a linear blend or a curve. Compose has an elegant interface that is easy and intuitive to use. Much like a nonlinear video editor, it's possible to scale, repeat, copy, and blend animations using visual representations of the clips on multiple tracks in a timeline.

You can also modify any aspect of the character's performance on an instance of the motion clip, or even modify the original motion clip itself. Compose is a well-refined NLA with many second-generation features. For instance, a walk-cycle motion clip captured from one character can be applied to a different character using a tool that matches the individual items in one group to the items in a different group. Most NLAs have similar features, but Messiah's has advanced features like auto-matching, hierarchy matching, and base-name filtering.

Another advanced feature is how Compose applies the edited motions to the character. For each item in a character group, you have the option to apply the motion before or after the character's IK. Motions can also be applied instead of — or combined with — the motions that the character has in the current scene already. The possible combinations are additive, subtractive, multiply, divide, greater than, or less than.

You can also have a variety of motion end behaviors (such as cycle or oscillate) and rest value compensation so your character won't pop backwards at the end of each walk cycle. Another great feature is that you can define multiple character groups with overlapping items. For instance, you can make a character group for an entire character, and also make a group for just the hand.

In short, with all its refined features, Messiah: Animate is a pretty serious nonlinear animation system. Messiah: Animate has an MSRP of $1,045; the upgrade from Messiah 1.5 is $395. PMG is making available a free downloadable demo version of Messiah: Animate 3.0 at www.projectmessiah.com.

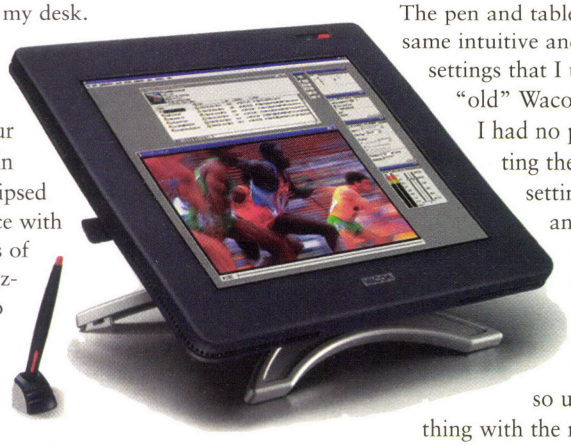
★★★★ | Messiah: Animate 3.0
PMG | www.projectmessiah.com

Sergio Rosas is an art director at Ion Storm. Contact him at srosas@ionstorm.com.

WACOM CINTIQ 18SX

by spencer lindsay

I've played with LCD pen tablets at trade shows before, but doing actual production work on Wacom's Cintiq 18SX is incredible — not that my coworkers let me get much work done when it showed up on my desk. Once the word got out that we had one of these down in our lab, every artist in the company traipsed through my office with nothing but cries of glee for this amazing tool. I had to wipe their drool off the monitor several times.



The Cintiq 18SX combines a high-resolution 18.1-inch LCD display with a pressure-sensitive pen tablet. The tablet sits on a cast-aluminum swivel stand so you can move your artwork to fit your position. If you're a lefty, it tilts the way you hold your regular pen and paper. I found that taking the tablet off of the stand and putting it in my lap like a sketchpad worked the best, but at 17 pounds (without the stand) it was just a bit too heavy to be comfortable for long periods of time. All the connection cables (the Cintiq supports both USB and serial and works with either a DVI or VGA video connection) are in one umbilicus, so there's no tangling. I can say that using the 18SX completely changes the way I interface with my computer.

Although I tend to like my monitor resolution way up in the "can't see it" range, Cintiq's 1280×1024 maximum resolution and 24-bit color depth are really great for Photoshop projects. After a bit of research, I was able to hook both the Cintiq tablet and my 24-inch monitor to the same workstation. Being able to work in Photoshop on the tablet and then switch to my 3D program and the mouse was a dream come true.

The response time of the tablet, rated by Wacom at 27 milliseconds, and monitor is fantastic. Being used to a bit of lag when working on larger images, I was pleasantly surprised to find the response time of the Cintiq was better than my current stylus. This had the effect of removing even further the interface barrier between my artwork and me.

The pen and tablet have the same intuitive and adjustable settings that I use on my "old" Wacom tablet, so I had no problem setting the pressure settings correctly and getting to work. Cintiq offers 512 levels of pressure sensitivity, so using this thing with the new brush modes in Photoshop 7 is amazing — just like painting with a brush.

The Cintiq 18SX includes a two-button grip pen with an eraser. The tablet is also bundled with Wacom's Pen Tools software and Corel Painter Classic, and Wacom offers an add-on kit for Irix-based workstations.

Simply put, this tool is revolutionary; not only does it simplify the computer for traditional artists, it introduces a whole new method of interface for us hardcore digital art geeks.

The Wacom Cintiq 18SX retails for $3,499. Go get one.

★★★★★ | Cintiq 18sx LCD Tablet
Wacom | www.wacom.com

Spencer Lindsay recently moved to Carlsbad, Calif., where he is currently working at Angel Studios.

Buy an engine. Gain a partner.



When you license NDL's NetImmerse 3D game engine and development toolkit, you get much more than great code. You gain a development partner with engineers dedicated to doing everything possible to ensure the success of your title.

NetImmerse is the only engine that can deliver all of this:

- A complete toolset that accelerates your art pipeline.
- Optimized performance across all platforms.
- Full source code for all libraries and tools.
- Responsive support from those who develop the engine.



Great software. Solid partner. Proven success.

NetImmerse

The fastest path from conception to completion

916.783.4380 • sales@ndl.com

ndl

The Elder Scrolls III: Morrowind courtesy of Bethesda Softworks.
Star Trek® Bridge Commander™ courtesy Paramount Pictures. ™, ®, & © 2002 Par. Pic. All Rights Reserved.
Dark Age of Camelot courtesy of Mythic Games.
Tennis Master Series courtesy of Microids.
Freedom Force courtesy of Irrational Games.

www.ndl.com

alexander brandon
translation: kaito okutani

Shooting from the Hip: An Interview with Hip Tanaka

In 1986, we in the U.S. were playing the Nintendo Entertainment System (known as the Famicom in Japan), and roughly a third of the music for its first set of games was written by Hirokazu "Hip" Tanaka, then with Nintendo Co. Ltd. He has written music for countless other games as well as designing the Game Boy Camera and Printer, and also scored the soundtrack to the *Pokemon* TV series. He now runs Creatures Inc., which makes Pokemon cards and developed the hit Japanese Game Boy Color title CHEE-CHAI ALIEN. Thanks to the translation assistance of Kaito Okutani and Hideyuki Shida, I managed to have a few words with this legendary composer.

Alexander Brandon. How did you get involved in music for games?

Hip Tanaka. I studied electronic engineering at university, but you can probably guess what it was like. My professor once summoned me and said, "I understand if you want to make a living with your music, but I just cannot allow you to graduate as an engineer like this."

Then, I found a newspaper ad from Nintendo for a sound engineer position. I went to the interview, and fortunately I was accepted. At the same time, though, my band was selected as a finalist in a music competition and I got a chance to debut as a pro. It made me think about my future, but I decided to take the job at Nintendo. The band, by the way, found a replacement and made their national debut.

HIP TANAKA: SOFTOGRAPHY HIGHLIGHTS

- DONKEY KONG (1980): sound effects
- GYROMITE (1985)
- DUCK HUNT (1985)
- KID ICARUS (1986)
- METROID (1986)
- SUPER MARIO LAND (1989)
- BALLOON FIGHT (1990)

AB. For METROID, how did you go about creating the music? Did someone give you graphics from the game and ideas for themes?

HT. I had a concept that the music for METROID should be created not as game music, but as music the players feel as if they were encountering a living creature. I wanted to create the sound without any distinctions between music and sound effects. The image I had was, "Anything that comes out from the game is the sound that game makes."

They allowed me to be in charge of the game's music. I even insisted that game designers change certain graphical concepts in the maps from my point of view. Indeed, I named all the maps. In Japan, METROID was released on a disk system. The Japanese version has one more voice of polyphony, and it sounds much better.

AB. What was it like working with the technology to create music for the original Nintendo games?

HT. There was a dedicated sound design tool available when the Famicom was introduced. It was common for most sound designers to use sound tools in the PC and convert the MIDI data into Famicom's sound data. But then I didn't use any sequencers specialized for music and sound. I always created my own sequencer and used assembly for the programming language. Being a programmer and a composer using my original program was a strong element of my uniqueness.

AB. Do you have a studio now that you use to create music for current games and Pokemon?

HT. My company is called Creatures Inc., and the former president is the executive producer of Pokemon. I took a job creating the music for *Pokemon* TV series in my spare time, before Pokemon gained today's popularity. At first, I composed the music for it almost as a joke and didn't take it seriously. But I was asked to continue composing music for the anime series. Nintendo didn't allow employees to work for other companies, so combining my personal reasons that I had then, I made the decision to leave the company.

Everybody thinks I am a dedicated music composer, but before resigning from Nintendo, I planned and developed Pocket Camera and Pocket Printer. I gave a presentation to [former Nintendo chairman and president Hiroshi] Yamauchi, and I drew all the images and even programmed the prototype game myself.

AB. What are your thoughts on making music and sound for games now, compared to in the 1980s?

HT. Today, I think it's common now to work more systematically, not relying on one single sound designer as I used to do, but coordinating several groups who specialize in each field.

However, what really doesn't change is that sound design is a human-to-human business, not like you are doing it for dogs or birds. Not only that, but you have to tackle gamers' feelings, which is something we've never been able to grasp completely.

AB. Do you have plans to create music for an album or CD?

HT. Buckner and Garcia published a record album called *Pac-Man Fever* in 1982. In it, they had music for the DONKEY KONG arcade version. This song starts with Donkey Kong's stomps as well as Mario's squeaking footsteps. I made both of the sound effects, and you can't imagine how happy I was when I first heard it. I was really touched! I haven't heard of any releases in the near future regarding my works.

Wait, there's more! Check out the unabridged version of this interview at Gamasutra.com.

A New Design Freedom

3Dlabs
Graphics Evolved
A CREATIVE Company

Introducing the World's First Fully Programmable Visual Processor



The sky's the only limit to your design creativity with Wildcat® VP—an astounding breakthrough in 3D workstation graphics. The industry's first fully programmable Visual Processing Unit (VPU) powers a family of accelerators that deliver unmatched productivity for professional OpenGL® and Direct3D®—based applications. But that's not all. With unrivaled flexibility, Wildcat VP is enabling a new generation of applications to extend the boundaries of interactive visual realism. Experience the ultimate in graphics design freedom today. Be ready for visual programmability tomorrow. Wildcat VP from 3Dlabs.

Leading OpenGL and Direct3D performance	A full family of 64MB and 128MB boards
Over 200GFlop and 1.2 TeraOp VPU	High performance virtual memory
Professional-grade reliability and quality	Dual independent displays

Wildcat VP

www.3dlabs.com

3Dlabs is a registered trademark of 3Dlabs Inc. in the United States and other countries.

Toward Better Scripting: Part 1

Every year we create larger amounts of content for our game worlds, and each piece of content becomes richer. As a result of this trend, scripting languages are becoming increasingly important. A scripting language should enable content builders to make game events happen easily by expressing ideas in a high-level manner. And ideally, a faulty script won't crash the game and disrupt workflow.

With subjects like graphics or AI, we often turn to academic research for solutions. This is a good idea for programming languages, too, but it doesn't work as well. Most programming language research doesn't have much to say about making events happen in a game world. Much of what we want is highly domain-specific, so we should apply our domain experience to come up with ideas for solutions that other people wouldn't consider.

It's difficult to devise good features for a scripting language; we don't have many good examples, since most familiar languages are made for a different kind of programming from the one we want. So in this month's column and the next, I'm going to concoct an experimental programming language feature. My goal is to show an example of a tool that could be useful for games that we would be unlikely to see in a general-purpose programming language. I also want to play a little bit with the feature and show the interaction between domain-specific concerns and language design.

Dealing with Time

I like 3D games, which are about events that happen over periods of time inside some region of space. Traditional programming doesn't deal well with time and space, so they should be good subjects to explore for domain-specific improvements.

We often want to talk about the behavior of things that change over time. We want to ask questions such as "Has the player been closely following the messenger he's supposed to spy on long enough to complete the mission?" or "Have there been enough combat events lately to keep the player interested, or should we increase the number of wandering monsters?" or "Is the player making continual progress toward the goal, or should we guide him more?"

Programming these aspects of a game is usually a lot of work. We need to make a piece of code that runs every frame and measures certain things about the state of the world, then records those values somewhere so that we can look at them in the future, or averages them together into some quantity about which we then make decisions. These systems often grow to be unwieldy, and they easily pollute other portions of the code base (we'll see some examples of contamination next month).

In most programming languages, when we assign a variable to a new value, the old value is completely erased. But if the language possessed some kind of short-term memory about how the variable has been behaving, we could use that facility to answer a lot of time-based questions. And maybe in the end, our scripting language will be a little more organic, a little less mechanical.

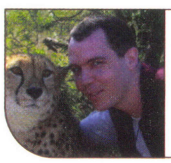
I will implement this memory by building it into the variable handler for a scripting system. Since it's in the core of the interpreter, we can invest effort to make the feature high-quality (for example, mathematically well defined and frame-rate independent), and the benefits will propagate through the system: everyone who performs time-related queries will get high-quality results.

I am choosing this feature because it's kind of cool and something I've never seen before, and it's a facility you could imagine existing in almost any language. I'm not choosing it as a magic bullet to solve all our scripting problems; time-related queries are not necessarily the biggest content-development bottleneck. But by exploring features like this, step by step, we can eventually arrive at a programming paradigm very different from what we have now, a paradigm that is much more productive for us.

How Variables Tend to Work

One of the main tasks of a scripting system is to manage communication between the core game engine (fast, low-level code) and the running scripts (slow, high-level code).

So that the scripting system can perform code execution and run-time type-checking, each variable accessible by the scripting language is stored in a table, and the entry for that variable is a data record containing more than just the variable's value: for example, a string that tells you its name and an integer telling you its type (Boolean, real number, integer, entity ID, and so on).



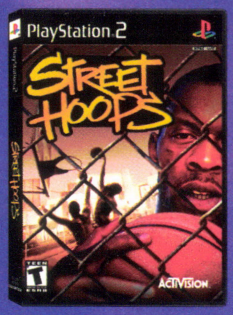
JONATHAN BLOW | *Jonathan has taken a renewed interest in programming language design. Send polemics about language semantics to jon@number-none.com.*

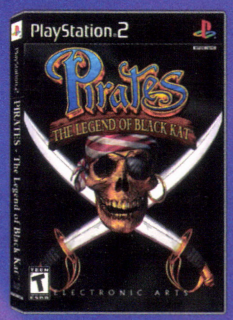
Don't Just Play It, Live It!™

SURROUND SOUND FOR PlayStation®2

ONLY FROM

dts INTERACTIVE





DTS DELIVERS THE ULTIMATE GAMING EXPERIENCE:

THE ONLY REAL TIME 5.1-CHANNEL DISCRETE SOLUTION FOR PLAYSTATION®2

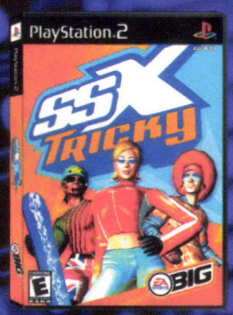


The world first heard the power of DTS® digital surround sound in movie theaters with Steven Spielberg's Jurassic Park. Since then we have dedicated ourselves to making ***everything*** sound real.

Now DTS delivers the theatrical experience of up to 5.1 discrete channels ***during game-play*** with DTS® Interactive for PlayStation®2 games.

Find out how the DTS Software Developers Kit helps developers like Electronic Arts make the game real.





For more information on DTS® Interactive SDK:
Glenn Arentzoff garentzoff@dtsonline.com
818-706-3525

www.dtsonline.com

Click on the Pro Audio button to access more information about DTS Interactive

Playstation is a registered trademark of SCEI, Inc.

I will store the history information inside this data record. History will automatically be available at all times for every variable in the system, without requiring the programmer to ask explicitly for variables to be tracked. Behind the scenes, the engine must be updating the history of all variables defined in the system, so the history update needs to be a fast operation.

The majority of these computations will produce values that are never referenced, and many optimization-obsessed programmers will grimace at that notion. But I want the history information to be easy to use and maximally dependable. The programmer should expend exactly zero extra effort to get at it. I don't want the unreliable behavior that would happen otherwise. Imagine that a script is dynamically loaded, and that script suddenly wants to know how a certain variable has been behaving for the past 30 seconds. But the information isn't available, because analysis of the previously loaded scripts told us that nobody would care about that variable's history. This is a specific case of a pattern I call "optimization impeding progress." We end up spending huge amounts of time debugging and not enough time actually making the game.

I don't want come across as an anti-optimization bigot; optimization is very important in creating smooth-running systems. But we always need to weigh the benefit of an optimization against the damage it causes, being cognizant of the house-of-cards complexity that we tend to build up after many optimizations. The talent in being a good modern software engineer is in choosing the few best optimizations, not in optimizing everything. We need to optimize our optimization efforts, as it were.

I expect that the amount of CPU we spend tracking history will be vastly overshadowed by time spent running actual script code. Optimization in this case would be unproductive.

Vectors Changing over Time

We could devise a system where, every frame, we sample the current value of some variable and store it in an array. Then scripting code could look over this array and make decisions every frame. But storing a large number of samples for every variable is costly, and it's unclear how we should process that sample array consistently and meaningfully (for example, without variable frame rate messing us up). We might be led toward methods involving signal reconstruction from nonuniform samples, a slow and painful activity.

So instead of recording many exact values, I want to record a small number of approximate values that tell us about the trend of a variable over time. As a result, the system can answer questions about the general behavior of a value but can't say precisely what the value was at a given time.

Some time ago, a coworker showed me a simple hack for smoothing rapidly changing values. If x is some fluctuating input value, and the ith sample of x is x_i, then $y_i = ky_{i-1} + (1 - k)\,x_i$, where $0 \le k < 1$ is a value that changes more smoothly. Another way to interpret "more smoothly" is "takes some time to converge to the input value," where the amount of time is controlled by k. (This equation is a simple low-pass IIR filter.) This construct originally came up when we were drawing frame counters. Instead of displaying the instantaneous frame rate, which is difficult to read because it changes so quickly, we would smooth the frame rate by two different values of k, to show short-term and medium-term trends.

If the language possessed some kind of short-term memory about how the variable has been behaving, we could use that facility to answer a lot of time-based questions.

This filter is much like what I want, except that the rate of convergence here depends on the number of iterations you run. So if you perform this operation once per frame, the rate of convergence depends heavily on the frame rate. But you can derive a less-hacky, frame-rate-independent version of this operation. Make k a function of duration, then write an expression that says the results of filtering serially, first with $k(\Delta t_1)$ and then with $k(\Delta t_2)$, should be equal to filtering once with $k(\Delta t_1 + \Delta t_2)$. Work some math and you get $k(\Delta t) = (b^a)^{\Delta t}$. Both b and a are parameters you can adjust, but for our purposes they comprise only one degree of freedom, as we'll see in a moment.

For each variable, I set up an array of these functions, where each function converges at a different speed. I'll control these by designating that h equals the amount of time it takes for the filter to converge to 90 percent of a fixed input value. Convergence is asymptotic, so the filter will never actually reach the input value. The values of h might range from, say, 0.1 seconds to 30 minutes.

The appropriate k is $k(h, \Delta t) = (0.1^{1/h})^{\Delta t}$, where Δt is the frame time. The division and exponentiation look scary, but actually our system can be very fast. The value of k depends only on h and Δt; h is fixed permanently, and the frame time changes only once per frame. So when it's time to update the scripting system, we precompute the handful of values of k that we will be using. Then we iterate over all the variables, updating their history data using only addition and multiplication.

ADVERTISEMENT



MAXimum Audio Intelligence

soundMAX SPX

Sound designers and programmers have long had many options for creating the audio for games, but with the increase in processing power and the addition of powerful specialized audio subsystems, the possibilities have increased dramatically in recent years. Long gone are the times when a simple system beep, or a single looping wave is adequate for creating the kind of immersive audio environment required for today's games. Yet, even with this increased performance, the system resources available for audio are still sorely limited. The challenge, as it has always been, is how to maximize audio quality and ease of programming within the constraints of the platform, and how to evaluate the tradeoffs involved.

The Sound Complexity Continuum

When evaluating sound generation options for a game, one methodology is to establish a priority for a given sound or sound type (is it a background sound that only occurs infrequently, or it spawned repeatedly by the player's character?), and then balance it with other required sounds along some kind of continuum. One possible way to order a continuum is complexity, which translates to complexity for both the sound designer and the programmer, and also translates directly into CPU cycles required. This continuum also provides an easy way to balance the resources required for the audio environment - background or infrequent sounds being served well by the leftmost methodologies (see Fig. 1), and primary game sounds requiring as advanced an approach as possible, to maximize realism and minimize repetitiveness.

Fig. 1



Real time DSP processing is available in hardware on some platforms, and is feasible in software on others, but in general the CPU requirements limit DSP usability on platforms that do not support it in hardware. In general, DSP post processing should be considered 'special sauce' - something that is good to have, but cannot be counted on in any consistent way for key sounds. The methodology chosen for a given sound should sound good without it.

Full Synthesis of audio in real time has been available for some time in various forms, and although it has the potential for extremely interactive sound with small memory requirements (some very realistic vehicle examples have been produced, for instance), current limits on CPU time and the availability of useful sound-design tools have limited its acceptance as of yet. Significant research and development opportunities still exist in the creation of useful and understandable tools for content creation and game integration using pure synthesis, to reduce the burdens placed on the sound designer and programmer, so that the full potential of real time synthesis can be realized by the average (or even advanced) developer.

It should be noted that pure synthesis in game context is not by any means a new idea. In the early days of game sound, using samples was not an obvious choice, due to memory cost and other factors. Although a few pioneers were able to coerce sample playback out of hardware not designed for it, most game sounds used various synthesis methods instead. These included ingenious software-based methods, such as the classic Williams sounds (Defender, Robotron, etc), which were implemented in software on a 6808 processor, to dedicated hardware, such as Yamaha's OPL FM synthesis chips. The desire for more realistic (and more easily editable) sound caused sample-based methods to overwhelm these methods soon after they became feasible.

With the simpler sample-based sound methods firmly established, and the "high-end" of pure synthesized sound being explored, but not yet feasible on many platforms, there is the upper middle ground on our chart to be filled in. The sounds here range from the more complex methodologies of pre-composed sample sequences to fully algorithmic sample sequencing. It represents an evolutionary step from the point of view of the sound designer, pushing the boundaries of the familiar sample paradigm rather than adding the whole new Pure Synthesis paradigm. One of the primary advantages of algorithmic sample sequencing is that it maximizes the power of the wave hardware on a given platform, without using too many CPU cycles on the primary processor.

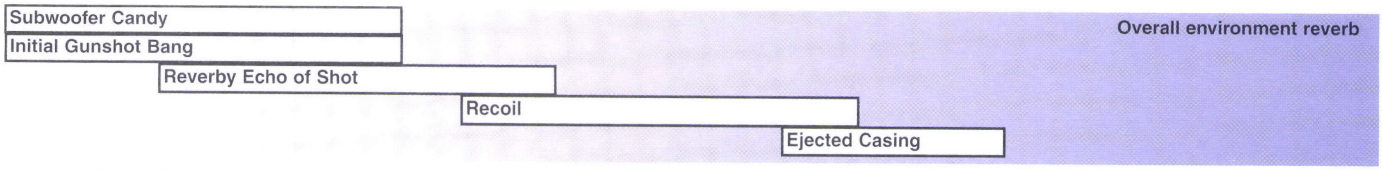
Pre-Composed Sample Sequencing

As an easy step up from simply layering or randomizing single-sample sounds, a pre-determined sequence of smaller source samples can be offset in time and layered to form a complex sound. At trigger time, the algorithm can choose among a set of random samples, to reduce the absolute repetition of a single sample. The algorithm can also assign parameters such as volume and playback rate (pitch), randomly chosen from values within designable ranges. With just a small number of layers and waves per layer, the number of possible wave combinations per trigger can expand quickly and greatly reduce the chance of audible repetition from trigger to trigger.

ANALOG DEVICES

For example, one could set up and outdoor gunshot sound as follows:

Fig. 2



Some layers may have only a few waves to choose from (subwoofer candy comes to mind, which might have only one wave in its list), others might have more. The combination of multiple layers plus algorithmic randomness would produce a complex sound that has an extremely low repeat likelihood, at significantly less memory usage than using enough full recordings to get a similar repeat probability. (It is important that no single wave in any group stands out enough that it is obvious when it is chosen, thus breaking the illusion of non-repetition). Furthermore, one might be able to add a level of interactivity to the sound by influencing the random choices of the layers. A simple example would be to vary the volume of the "subwoofer oomph" layer according to some "power" parameter. Other parameters could be varied, such as probability that a layer triggers at all, or what part of the layer's wave list to choose from. One could order the "echo" waves according to closeness of the surrounding geometry, and weight the random wave selection according to location of the player in the environment, or one could order the "bang" waves by several gun parameters (heat build-up, wear and tear, amount of ammo), and weight the randomization according to that parameter.

With some thought, this type of layering could be applied to sounds with less obvious sub-sounds for each layer. Say, for example, that one wanted a simple gunshot that was the equivalent of Layer 2 (Fig. 2), the initial gunshot. There are a number of ways to split the source sounds and recombine them during playback to achieve variations on the original sound. Splits might be in frequency, for example, or in finer time sections, or one may simply decide on layering full unsplit waves.

Fully Algorithmic Sample Sequencing

The concepts of randomization and layering can be extended, via an analogy to the concept of particle systems in graphics. In a graphics particle system, a set of particle primitives is spawned by an emitter, with randomized properties (size, intensity, color, velocity, location, etc), and randomized start times. Similarly, in an audio particle system, a set of "particles" (bits of sound) is spawned with randomized properties (volume, pitch, wave choice, etc.), and random times. There are various ways of defining a particle in such a system; one straightforward choice would be "an instance of a wave." Other choices can be found in the field of Granular Synthesis, which on the whole can be seen to be highly comparable to the graphical field of particle systems (as synthesis methods, granular techniques form a continuum of their own, straddling the gap between algorithmic sample sequencing and Full Synthesis, and offer a ripe field for exploration in game sound --- particle systems as described here only touch the tip of the iceberg). With this definition of a particle, we can come up with a particle system method that is basically a random sample sequencer, extending the previously covered concepts of randomization and layering nicely.

Particle systems tend to fall into two groups: Particle Bursts and Particle Flows. Bursts are systems that sound for a finite length of time then stop, whereas flows continue indefinitely. Flows and Bursts can be thought of as the particle-system analogues of looping and non-looping waves. Particle bursts allow for the non-repeating simulation of various events that have particle-like properties, such as the sounds of breaking glass, explosions, falling rocks, footsteps in gravel, etc. Particle flows, on the other hand, can be used to implement continuous sounds without loop artifacts, particularly ambiences such as audiences, birds in a forest, water dripping, etc.

It is often useful to combine particle systems with simpler sample-based techniques, such as wave randomization, in cases where full particle simulation of the full soundscape isn't desired. For example, it has turned out to be common for a particle flow to be "filled-in" with one or two background loops of base ambience, which would be too much work to turn into particles, or may just be inappropriate to do with particles (such as background rumbles, murmurs, etc.).

Using the Right Tools For the Job

For a methodology such as algorithmic sample sequencing to become widely accepted, two factors must be satisfactorily addressed: sound designer burden and programmer burden. The challenge here is twofold: creating algorithms that can be used in as wide a range of applications as possible to minimize recoding, and the creation of tools that allow the sound designer to realize the full power of the algorithm without over-complication. The simplest sample-based methods had relatively little sound designer burden, in that the techniques were essentially those already in use in other fields, and the programmer burden was satisfactory. Still, sound programming toolboxes for even the simplest methods are widely used, as they minimize these burdens as much as possible. More complex methods need similar support to become useful to the general game development community. As the concepts become more commonplace, more toolkits become available to support them. One such toolkit, Analog Devices' SMartTools - including SMartAnimator, SMartSynth and associated runtimes for various platforms, contain tools for designing and using complex sound types, including sounds of both algorithmic sample sequencing and pure synthesis methodologies. The tools are intended to provide powerful algorithms for creating less repetitive, more interactive sound within the comfortable sample paradigm, with as little a learning curve as possible, for both the sound designer and programmer. In summary, as computing environments continue to become more and more powerful, developers of games now have the ability to use increased processing power for hyper-realistic audio, without impeding the other elements of game play. By perfecting algorithmic technology used to render audio, and by incorporating it into the application code of a game on a PC or a console system, we have now reached a point at which software designers can significantly enhance the realism and impact of the user's experience with a heightened audio experience.

For more information please visit: www.audioforgames.com

Once this mechanism is in place, we can ask questions about a scalar value, such as "Has it been generally increasing or decreasing over the last n seconds?" just by looking at the history data. You can imagine abstracted language primitives such as `Trend(variable, time_period)` that might encapsulate this concept. We can also ask roughly, "What has the average value of the variable been over the last t seconds?" though it's actually unclear what a perfectly well defined notion of average would be; you get into signal-windowing issues. But our version is pretty good.

To answer questions about a variable for arbitrary times, we need to interpolate between the history values. For now, I'll just use linear interpolation; maybe next month I'll discuss alternatives.

We can straightforwardly extend all this filtering to vector values by performing the scalar history tracking on each coordinate of a vector. So we can also talk about the history of where entities have been in space.

Fluctuation

It would be useful to know more than the average of some particular value. We might want to ask, "Is this value steady or is it fluctuating? Is it becoming more stable over time or less stable?" We can answer these questions if we track the variance of our variables. We blend variances together over time, the same way we are computing means.

Variance of a scalar variable is another scalar, which you compute by averaging the squares of the input values. The variance of an n-vector is an $n \times n$ matrix, which you compute by averaging the "squares" of the vectors, which you get by taking the outer product of each vector with itself. (I have put "squares" in quotes here because there are several valid ways of multiplying vectors, one alternative being the Clifford product). For more detail about this outer product matrix, see last month's column, "My Friend, the Covariance Body" (Inner Product, September 2002). I called this a "variance-covariance matrix," among other things, which is what you should search for on the web for more information. In general, though, I prefer to think of this matrix as being just the variance of a vector variable.

The matrix is symmetric, so we only need to store the upper triangle, which is nice. Eigen-decomposition of the matrix yields an ellipse that tells us generally where in space a point has been. So we can use this to ask questions such as "Has the player been exploring a wide area, or has he been sedentary? What's been the primary direction of his travel, and has he been mostly going straight or has he zigzagged a lot?" You might think to answer these questions just by analyzing a log of averages of the player's position, but you run into aliasing problems that way. The variance carries information of a rich character that can be leveraged effectively using a small number of samples, as you can see by running this month's sample code.

Last month I presented the covariance body as an encapsulated concept, but sometimes it's better to violate this abstraction. Recall that in order to combine two variance matrices, we need to shift them so that the variances are computed around a common point. Often this common point is the mean value of the inputs. But to make the system as simple and fast as possible, I have chosen to compute the variances of all values with respect to the origin. So we can just add the matrices together without shifting them. When the time comes to display a variance, then we compute the shifted matrix and perform the eigen-decomposition on that.

Sample Code, and Next Month

This month's code, available at www.gdmag.com, defines two variables that get updated in real time: one scalar, representing the clock; and one 2D vector, representing the position of the mouse pointer in the window. Once per frame, a history manager computes new averages and variances for these variables. Each frame, the histories are drawn on the screen: a linear trail of white showing the pointer's average positions and a series of blue ellipses showing the typical areas covered. Pressing *s* will toggle an enforced slow-down of the frame rate; you can see that the behavior of the averaging stays consistent despite the frame rate changes.

Aside from the variable declaration system, there's no actual scripting language yet. I'll fix that next month, when I look at some operations we can support on history-augmented variables. We'll also examine some concrete applications of the history feature. There's no room to discuss the examples fully in this article, but in order to end with something tangible, I'll summarize them now.

The first example is a commentator for a game like DANCE DANCE REVOLUTION. The game assigns you a momentary rating based on how well you hit each arrow; this rating fluctuates very quickly. But because the rating is automatically being tracked over time, the script can easily check for broad patterns in the history. So if you do poorly for a while but then start doing very well, the commentator might say, "What a comeback! For a while, I thought you were going to lose!"

The second example is a mortar unit from a real-time strategy game. It fires ballistic shells that take some time to reach the target; in order to hit, it should only fire when the target is staying within a relatively small area, though the target doesn't have to be unmoving. Imagine a soldier alternating between two positions in a trench. Since every unit in the game has a position vector, and that vector is being remembered over time, the mortar vehicle can check how spread out the target's position has been over the past n seconds and fire if the result is tight enough.

Next month I'll look at both these cases, see how they might be done traditionally, and then implement them using history information.

Character Matters: Part 2
Getting Ahead of the Curve

Art at my school was a compulsory subject until you turned 14. When you turned 14, you had the option to continue by choosing it from a group that included metalwork (how to burn yourself badly with a welding torch), domestic science (how to burn yourself badly with a pan full of boiling custard), and woodwork (how to lose most of your fingers in the band saw). If, like me, you stood a better-than-average chance of getting permanently maimed in any class that involved rotating blades or white-hot metal, art seemed like the only route to graduation with all my extremities intact.

Unfortunately, my high school had been around for just over 100 years and considered itself to have a proud tradition of academic and sporting excellence. Its reputation was essentially built on two things, its rugby team and its absolute refusal to move with the times.

In practice, this meant that all fundraising and additional budget in the school went to send the rugby team and a selection of teachers to Australia each year to play schools from the Southern Hemisphere. In addition to the funding wasteland that was left in the wake of my school's pursuit of sporting excellence, its refusal to modernize either teaching methods or equipment meant that art lessons were a little bleak.

In addition to the appalling state of the school's resources, teaching methods that hadn't budged since the era of Jack the Ripper meant that in six years I drew more bowls of fruit than you could possibly imagine.

As our school was barely able to find the budget for a few dozen pencils, computers were something that only existed in the outside world. During my last years of high school, the Commodore Amiga emerged as an affordable machine with unprecedented graphical capabilities (4,096 colors was almost beyond comprehension for those of us who had come through the 8-bit scene). Meanwhile, my art classes remained fruit-centric, even though I was now at least using oil pastels.

A few years later, as the first modeling packages available to anyone outside of Pixar began to emerge, a whole three-dimensional world started to unfold. It became apparent that for those who were interested, a new set of skills needed to be developed along with those that helped create an image in the flat world.

In a Mary Poppins, supercalifragilisticexpialidocious sort of way, the full explanation of NURBS sounds more grandiose than it actually is.

As time went on, the tools evidently became more powerful, and we now find ourselves in the present, where there is little that a high-end package can't accomplish in the hands of someone who knows what he or she is doing.

It's all very well to be able to push a pencil around, and rules of proportion, perspective, and composition are all good to know, but if you work in games and are expected to create a world filled with characters, you'd better be able to apply that knowledge in all three dimensions.

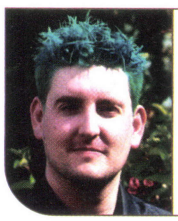
Take Your Pick

Fire up any search engine and type something like "3D character modeling," and you will be faced with an interminable list of forum discussions and tutorials about the best way to build a character. Across the various packages available, there are a number of basic methods that crop up, each with its own advantages and disadvantages, supporters and detractors. Evidently, building a character for a game is not quite the same as building one for a Disney movie (not yet, at least), but there are still a variety of options worth considering. The following are two of the most popular.

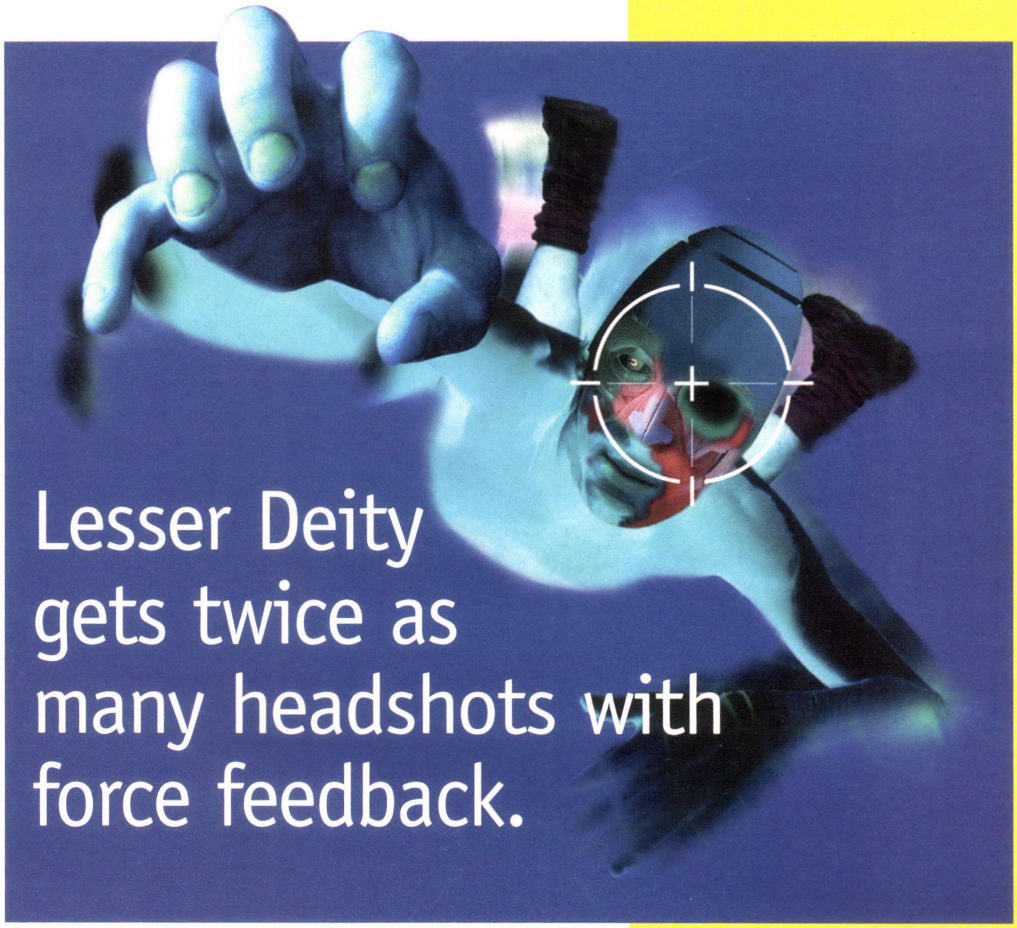
NURBS

If you find that your father, like mine, is still waiting for you to get a proper job, preferably something that involves either some form of mathematics or moving around heavy objects, try talking to him about your close association with Non-Uniform Rational B-Splines. In a Mary Poppins, supercalifragilisticexpialidocious sort of way, the full explanation of NURBS sounds more grandiose than it actually is.

In their full glory, NURBS are basis-spline curves described by an equation expressed as a ratio between two polynomials, created by the interpolation between three or more points, along



HAYDEN DUVALL | *Hayden started work in 1987, creating airbrushed artwork for the games industry. Over the next eight years, Hayden continued as a freelance artist and lectured in psychology at Perth College in Scotland. Hayden now lives in Bristol, England, with his wife, Leah, and their four children, where he is lead artist at Confounding Factor.*



Lesser Deity gets twice as many headshots with force feedback.

"Force feedback allows me to pinpoint a target and go in for the kill with better precision."

Lesser Deity,
#1 ranked in
Unreal Tournament
ngStats May 2001

When done right, force feedback technology lets you aim better, fly farther, drive faster, and generally play better. Let Immersion show you how.

Immersion's developer tools are designed to get the most out of the Microsoft DirectX force feedback API. They make it easy to:

- Choose from hundreds of pre-made "clip art" effects
- Design custom effects using What-You-See-Is-What-You-Feel authoring tools
- Experiment interactively with effect parameters

Read success stories at
www.immersion.com/developer

Contact us to learn more or to receive our developer's kit:
1-888-467-1900
developer@immersion.com

Immersion
TouchSense Technology

Immersion Corporation | 801 Fox Lane, San Jose, CA 95131 | (408) 467-1900 | www.immersion.com

Copyright 2002 Immersion Corporation. Immersion, the Immersion logo, and TouchSense are trademarks of Immersion Corporation.
All other trademarks are the property of their respective owners. All rights reserved.

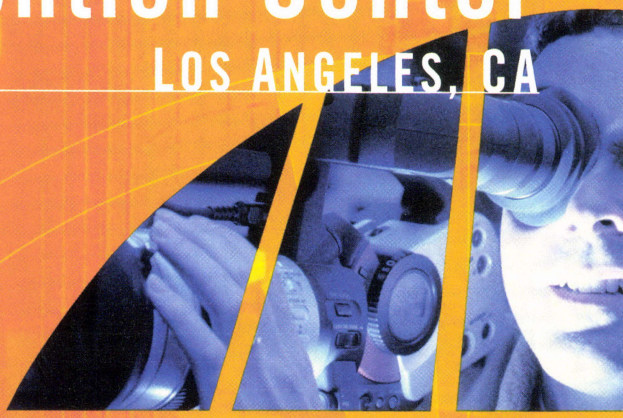
INSPIRING AND EMPOWERING CREATIVITY

Digital Video DV expo WEST

DVexpo.com/west

Los Angeles Convention Center

LOS ANGELES, CA



CONFERENCE: DECEMBER 9-13, 2002

During five days of in-depth instruction from the industry's hottest talent, the **SIXTH ANNUAL DV EXPO WEST** conference teaches the skills and techniques to shoot, edit, and deliver the most captivating digital video content.

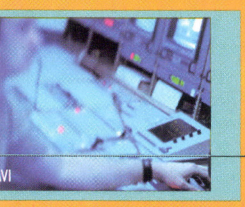
NEW! DV Certification

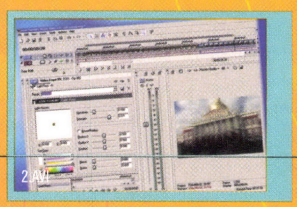
New to this year's curriculum is the chance to earn DV CERTIFICATION through hands-on courses in Final Cut Pro and Avid Xpress DV, taught by certified trainers.

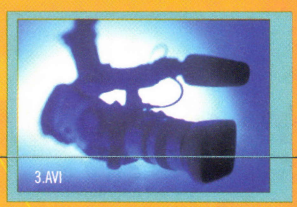
EXPO: DECEMBER 10-12, 2002

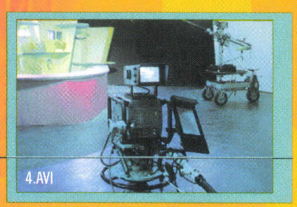
Network with peers, browse the latest gear, and get a sense of technologies that are revolutionizing the digital video industry from companies like Adobe, Avid, Canon, Canopus, Discreet, JVC, Matrox, Mole-Richardson, Panasonic, Pinnacle Systems, Sony, and many others.

Register online at www.DVexpo.com/west with source code: DVSEPT







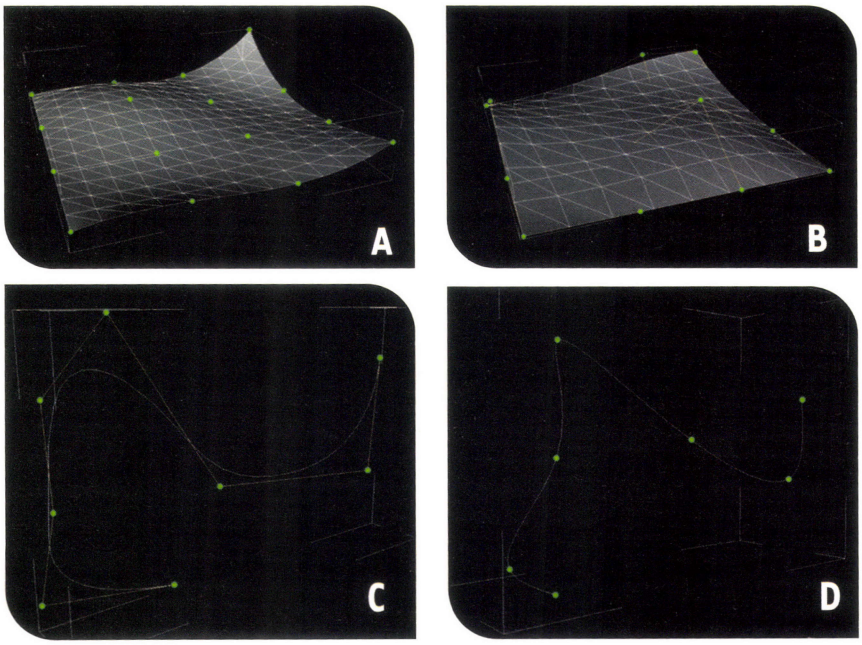


CMP
United Business Media

Digital Video DV Media Group

Images provided by CREATAS from the Digital Vision, Thinkstock, and Stockbyte Libraries

AGD



FIGURES 1A–1D. Using NURBS gives artists the option of using point surfaces (1a) and point curves (1c) to manipulate points that lie directly on the surface or curve. CVs, or control vertices (1b, 1d), offer artists an alternate interface.

which the extent of the influence of its control vertices may vary.

If you look into the description of NURBS objects, apart from developing a headache, you will find that these awkward little guys exist in parameter space as well as regular geometric space, and that their mathematical origin is both a plus and a minus. You can also edit NURBS using point curves and surfaces as opposed to control vertices (CVs), which is essentially just an alternate interface, as the underlying NURBS are still constructed using CVs. A point curve or surface, however, allows the user to manipulate points that lie directly on the surface or curve. It does not rely on weighting to define the magnitude of its influence but rather its positioning. Figures 1a–1d show how NURBS and CV point curves and surfaces differ.

So what, in practice, do we do with NURBS that lets us turn all of this terminology into a character?

One of the most popular methods of NURBS modeling is an approach that is similar to building a wire framework over which a surface will be applied. In this case, the framework is one of splines, and the surface will be generated over these using each spline's shape as a guide.

The advantage of NURBS is primarily that you can create organic, smooth surface contours quite easily and avoid the angularity of polygons. NURBS also allow you to adjust shapes through the manipulation of CVs, all of which (through position and weighting) can produce smooth changes across the surface, retaining its organic feel. Effectively, this means that the number of control points that you need to consider for a very smooth mesh is far lower than would be necessary if the model was simply made from polygons (with the associated number of vertices).

The NURBS naysayers bring up a few downsides that are worth considering. For one, the system overhead is larger with a NURBS model than the much more efficient polygon geometry.

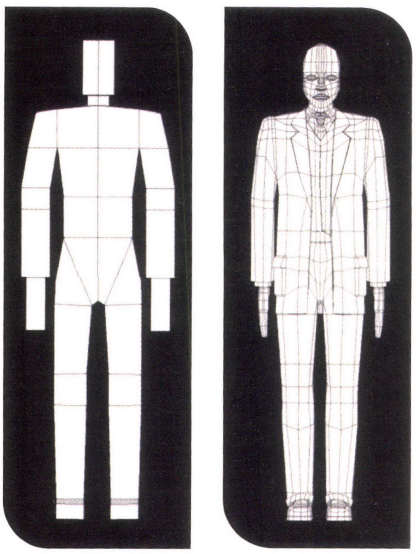
But one of the main problems with NURBS is that after a couple of days trying to model a character, it is easy to feel that you've been involved in protracted negotiations with your software. Modeling with NURBS can sometimes be a bit like modeling through a third party; in this case, the third party is the mechanics behind NURBS construction. A common complaint about the NURBS approach is the perceived distance it creates between the artist and the final product. The manipulation of a NURBS object doesn't provide the direct interactive feeling of working with a form, and adding detail exactly how and where you want it can be anything but straightforward.

In addition to these problems, animation with NURBS carries a much larger overhead than animating a polygon mesh, so ultimately a NURBS character is likely to be converted to polygons after it has been built. This top-down approach creates a high-resolution mesh with the understanding that it will be optimized to a lower level of detail once the modeling has been completed. Currently, low-end characters are bottoming out at 1,000 to 2,000 polygons, mid-range characters are in the 3,000- to 5,000-polygon range, and some games are putting as many as 13,000 polygons into their characters at their highest level of detail. With these kinds of figures becoming the industry norm, a top-down approach such as NURBS is more viable than ever.

The Power of a Box

What is the antithesis of the mathematically intricate curve network at the center of NURBS construction? How about a plain old box? There aren't many objects less complex than a box, and this six-sided primitive shape forms the basis of another popular method for creating characters, subdivision surface, or sub-d, modeling.

In recent years, packages that deal specifically with this technique have arrived, prompting the heavy hitters of the 3D market to expand their tool sets to provide the same level of features. There is still a divide between the specialists and the high-end packages that cover a far greater breadth of functionality, but as "lower-end" graphics begin to be important for a growing number of peo-



FIGURES 2A–2B. A low-detail mesh (left) becomes a well-defined character through a subdivision of surfaces (right).

ple using these packages, the tool sets will continue to expand in the direction of subdivision surface modeling.

As far as character creation goes, the methodology is simple: it is the application that makes the difference, as is often the case. It's essentially a bottom-up process, and the basic procedure is as follows (the tools you need are available in just about any 3D package):

Step 1. Make a box.

Step 2. Edit the box until it forms a rough approximation of the shape you desire: extruding, scaling faces, moving vertices, and so on. At this stage, the mesh is very low on detail, and therefore it is an ideal time to concentrate on describing the basic volume that you are looking for without getting too hung up on the details.

Step 3. Refine the mesh until it meets the level of detail that you require. The first method is simply dividing edges where needed, working with the mesh right down at a basic triangle level, until you are happy. This method requires a good deal of vertex moving, and as the level of detail increases, so does the number of vertices that you need to manage. The second method is to use a smoothing function on the mesh to add the majority of the detail. Figures 2a–2b show an example of how subdividing surfaces turns a simple character into a more complex figure.

The exact approach each package uses varies slightly, but the procedure is essentially the same. A relatively low-detail model is used as a control mesh from which the computer is able to generate a more detailed version through a subdivision of faces. The exact structure of the control object then becomes less important in terms of its own shape, and more relevant in terms of the end result that is obtained through smoothing.

The specifics of this stage of the process have been covered time and again in many tutorials that deal with this method of modeling, but essentially, you need to be aware that the actual placement of vertices on your control mesh will be moderated by the smoothing function. Some features will need to be exaggerated, and sharp variations on the surface will be smoothed out unless edges are doubled to preserve the detail.

Also, the smoothing function is most reliable when the control mesh is created using four-sided polygons (quads). Maintaining as many quad faces as possible is often the most difficult part to master. With practice, however, the process becomes easier.

The sub-d faithful have long been advocates of this particular method of creating geometry, as it allows the artist direct control over every aspect of the mesh. This lets you add detail where you choose without any need to consider the effect it will have across the rest of a model's surface. It also is generally regarded as having a closer link to traditional sculpting techniques. The downsides are the large number of vertices generated at high detail levels and the need to break down the angularity of the triangles to achieve organic shapes.

The Way to Go

Looking at these two methods, how do you determine which is best for your project? It depends on whom you talk to and what you want to achieve. Here are some aspects of character design to consider when deciding which method to use.

Shape. If your character has many smooth curves and rounded forms, then NURBS modeling may be the right choice. However, at high levels of detail, polygons are obviously going to require the manipulation of large amounts of vertices, and maintaining smooth curvature can become an issue.

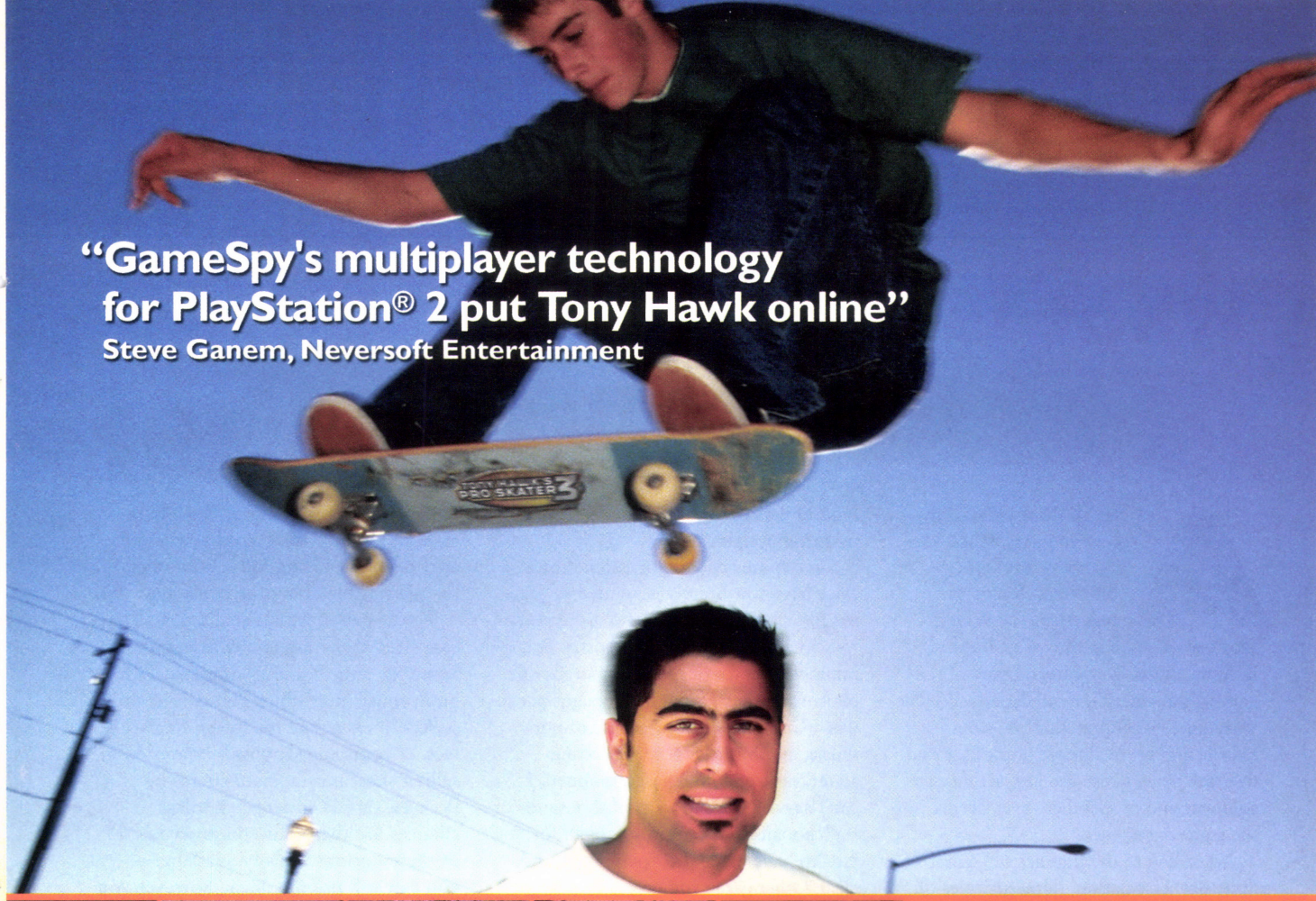
Flat, angular forms are perfectly suited to polygonal construction, and there is little point wasting energy on NURBS if your character is Boxalon the Cube-shaped Cube Monster.

Level of detail. As I mentioned earlier, NURBS modeling is more suited to continuous, smooth forms than those with abrupt surface changes, so if you design your character with a large amount of localized detail that needs to be part of a single mesh, polygons could be the way to go. As an example, a Giant Squid Monster might well be suited to a NURBS approach, but if it is a Giant Cyborg Squid Monster, adding in the mechanical details at the end of each tentacle would be a lot more straightforward if you use polygons.

Tools. Make sure your tools are suitable for the method you intend to use. The latest iterations of all the big players in the 3D arena have moved to cover both NURBS and the polygon approach, with varying degrees of success. There are also some relatively cheap packages on the market that specialize in the subdivision approach that are worth considering.

Changes/additions. Within reason, are there likely to be many changes or additions to the character's mesh? If so, adding detail to a NURBS model, or adapting it in some way, can pose more of a problem than working with a polygonal mesh.

The bottom line, however, is that a character's strength lies primarily in the quality of its conception and design, not in the method used to produce it. But it is good to know that there are a variety of options available to artists for character creation, and that it's all about choosing the method that fits best.





Tony Hawk's Pro Skater™ 3
for **PlayStation 2**

Where Middleware Meets Big Air

When Neversoft® and Activision® set out to create **the very first online game for the PlayStation 2** computer entertainment system, they turned to GameSpy.net to answer their multiplayer needs. GameSpy has over six years of experience designing rock-solid, easy-to-integrate online middleware for every major gaming platform.

But technology is only the start. GameSpy's reliable service keeps a game online for years by handling servers, support, and even bandwidth costs. We call it GameSpy.net: a complete turnkey online solution that delivers no matter what platform your game is being developed for. Call today to catch some big air of your own.

Find out more:
www.gamespy.net
bizdev@gamespy.com
Art Santos (949)798-4267

GameSpy is an official PlayStation 2 middleware provider.

- Peer-to-Peer Player Matching
- Server Browsing
- Instant Messaging and Buddy Lists
- Text and Voice Chat
- Online Storage of Game Data
- Competition & High-Score Systems
- Online Anti-Piracy
- Usage Analysis and Reporting
- Cross-Platform Networking
- Auto-Patching
- Online Data Transfer

gamespy.net™

Technology to put your game online.
Service to keep it there.

Copyright © 2002 GameSpy Industries, Inc. GameSpy and the SpyGuy logo are trademarks of GameSpy Industries, Inc. © 1999-2002 Activision, Inc. and its affiliates. Activision is a registered trademark and Tony Hawk's Pro Skater and Pro Skater are trademarks of Activision, Inc. and its affiliates. Tony Hawk is a trademark of Tony Hawk. All rights reserved.

What Is Your Game Music Worth?

After all these years in the game industry, it still surprises me that some game companies don't seem to realize that, by default, they own a stake in the music business. As in films, music is an integral part of the gaming audience's overall experience. The story line, visual presentation, sound effects, and musical score work together to draw players into the interactive environment and propel them ever deeper into the game experience.

According to the Interactive Digital Software Association, domestic sales of computer and videogames in 2001 totaled $6.5 billion. That's almost 225 million pieces of software sold last year in the United States alone. Although estimates vary, some industry pundits project that worldwide annual revenue for video and computer games will reach as high as $100 billion within the next decade.

Those of us creating these games are an important and powerful presence in the entertainment world, and we shouldn't discount the impact our products have on today's pop culture, and tomorrow's. Part of what went into the 225 million boxes snapped up and devoured up by eager consumers last year includes the music. It stands to reason that real opportunities exist for game companies to reap great rewards from music, simply by thinking a bit more like music companies.

Historically, film studios and record companies understand how music can generate magic when it comes to profit margins. Likewise, game music can generate revenue in many ways. For instance, was your game advertised on television? Was music from the game used in these ads? Was paperwork filed to the performing rights agencies reporting the play list of the ads? If not, your company may have missed potential opportunities to redeem tens of thousands of dollars in performing rights fees.

On a more general note, soundtrack sales have consistently generated revenue for film companies throughout the years — why not for games? At Artistry Entertainment, we routinely get hundreds of e-mails from fans saying they bought particular games specifically because of our music, and bushels more requesting soundtrack-only purchases. Granted, I can't say we get hundreds of thousands of such e-mails, but certainly enough for our fans to have made their point loud and clear. Other game composers I know are experiencing similar feedback. Spend some time in any game forum, and you're likely to see a number of lively debates on the merits of any number of game soundtracks. While I'm always pleased to find people enjoying and appreciating my music, the occasional criticism doesn't bother me much because it means above all that people are paying attention — and that to them, the music matters.

So how can you, as a game developer or publisher, use music to help make your game more successful? First, be sure your music has entertainment value on its own merit. Turn off the monitor, remove the gameplay, and see if the music still grabs you. If it doesn't — Houston, we have a problem.

Second, you have to think like a music company. Investigate every potential angle for generating revenue. Query professionals in the music industry, and research books written on the subject. Hire an expert consultant. Get advice from publishing and record executives, attorneys, marketing folks, and others knowledgeable about the music business.

Third, you must cut smart deals. You can often make money by saving money. Sometimes the simplest thing to do is grant music-publishing rights to composers in exchange for lower creative fees. If your company isn't interested in selling sheet music, managing sync licenses, or CD soundtrack sales, chances are that your composer has all of these options already available — you can still use the music however you want in your game, often in perpetuity. Composers with a personal stake in a score's financial performance always deliver better music.

And finally, be sure to review your contracts. Many companies are reluctant to deviate from boilerplate templates developed in an outdated industry and business environment. Companies slow to adapt to the new marketplace conditions concerning music may very well be truncating a wellspring of potential growth.

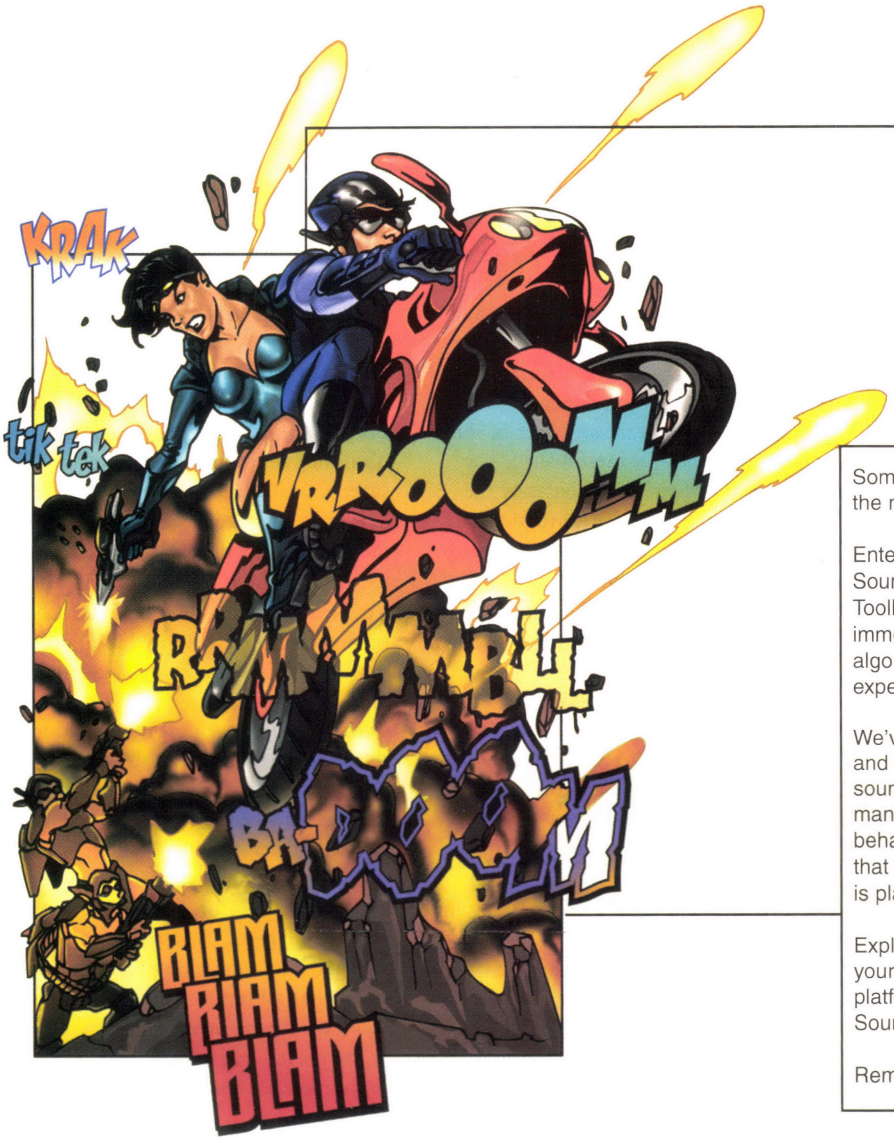
Plenty of marketing people in games have historically said that game music doesn't sell. Wrong — it is selling your game. If the music in your product isn't worth a stand-alone $10 to $15 soundtrack purchase price, then it's not as good as it should be, and definitely not as good as it could be. If it cannot compete in today's market, it's likely hurting sales — and your company's bottom line.



JEREMY SOULE | *Jeremy is one of today's most widely recognized game composers. He and his company, Artistry Entertainment, have created multiple award-winning soundtracks for the interactive entertainment industry's top publishers and developers.*

FORGET EVERYTHING YOU'VE HEARD.

SOUNDMAX SPX



Someday, you'll look back and remember the moment sound changed.

Enter SoundMAX SPX, powered by the SoundMAX Audio Rendering Technology Toolkit (SMartTools). Create ultimate, immersive audio, using advanced audio algorithms to deliver hyper-realistic game experiences.

We've blown through the sound barrier and warped into the next generation of sound dynamics. Now you can sculpt, manipulate and control complex audio behaviors. Form real-time sound events that you animate on-the-fly as the game is played.

Exploit the power of SoundMAX SPX in your next project. Fully scalable. Cross platform. It's your choice. Get SMart. Sound better.

Remember where you heard it first.

WWW.AUDIOFORGAMES.COM

SoundMAX SPX Partners

FOX INTERACTIVE™ · VALUSoft® · moto1.com All Motorsports All The Time · TESSERACTION GAMES · INTRINSIC · VICTORY SIMULATIONS · iMAGiNE interactive · Whispering Tree STUDIOS INC · DANGER PIRANHA GAMES INC · DSI Delphine Software International

SoundMAX® is a registered trademark of Analog Devices, Inc.
All other trademarks and registered trademarks are property of thier respective owners. All rights reserved.

ANALOG DEVICES

Begin at the Middle

"Where shall I begin, please your Majesty?" she asked. "Begin at the beginning," the King said, gravely, "and go on till you come to the end: then stop."

— Lewis Carroll, *Alice's Adventures in Wonderland*

"Let's start at the very beginning, a very good place to start."

— Oscar Hammerstein II, *The Sound of Music*

"Carroll, Hammerstein, what do they know about games?"

— Noah Falstein, "Better by Design"

The Rule: Begin at the middle.

When you are setting about to develop a game, rather than starting with the first level or initial scene of a game, pick a representative point near the middle and start there. The best order to develop a game is middle, beginning, then end.

The Rule's domain. This rule applies to game development processes for all types of games, but particularly those composed of discrete levels or chapters.

Rules that it trumps. "Start with what you know." It's tempting, and seemingly reasonable, to begin with the part of the game you know best, and often that's the start of the game. But there are two important reasons to start at the middle instead.

The first reason is that the beginning of a game is a special case, and so is the end, so both are liable to exceptions. By starting in the middle, you are likely to encounter more typical problems and opportunities, with a better balance of gameplay.

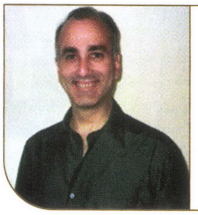
The second reason to start in the middle is that the first level you implement in a game is often cruder and less polished than those you implement later. But you don't want the first thing the player sees to be the crudest. Likewise, you want to leave the player at the end of the game with a sense of exhilaration and excitement, so doing the last levels first is unwise, not to mention impractical. Of course, you don't want any part of the game to be crude and unfinished, but reality intrudes and some parts of the game are going to be better than others. The middle of the game is encountered by players after they have already been hooked and before the game builds to a climax, and so is the safest place for less polished levels.

Rules that it is trumped by. "Do the hardest part first." Often, it is more critical to tackle the hardest part of the game, even if this is not in the middle. I'll have more on this rule in a future column. It also can be trumped by the rule "Throw away the first level you do." When the central activity of a game is uncertain, as can happen when you tackle an unfamiliar genre, use new technology, or incorporate many innovations, it can pay to implement whatever part of the game you understand best, and realize that later in development you will go back and throw out that first attempt when you have a better handle on the core gameplay. In this case it can still be helpful to start in the middle, but since you're going to redo that first effort anyway, it's not as critical.

Examples and counterexamples. While developing SINISTAR, my first published game, we tried to tune the turning and movement speed of the central player-controlled spaceship based on the first, easiest level. Then, when we had introduced a more crowded and busier later level, we found that the speed we had chosen was too slow to respond to the multiple enemy threats, and had to readjust. If we had started with the later level, or at least held off tuning until that point, we would have saved valuable time.

From the mailbox. Some readers sent me some compelling cautions about the use of asymmetrical distribution, the subject of my August 2002 column. Chris Bateman of International Hobo pointed out that although it is a useful rule, it is "somewhat offset by higher QA costs," since it is harder to test rare, random responses. And Charles of Chasm.org pointed out that my example of using it for standard responses in an RTS was flawed. Players count on unambiguous acknowledgments of basic inputs to let them know their orders are being carried out. He suggests, "Maybe you need another rule, 'Input Feedback,' which states that all input actions in the game will have direct, unambiguous feedback to the player, and which trumps 'Asymmetrical Distribution.'" A point well taken.

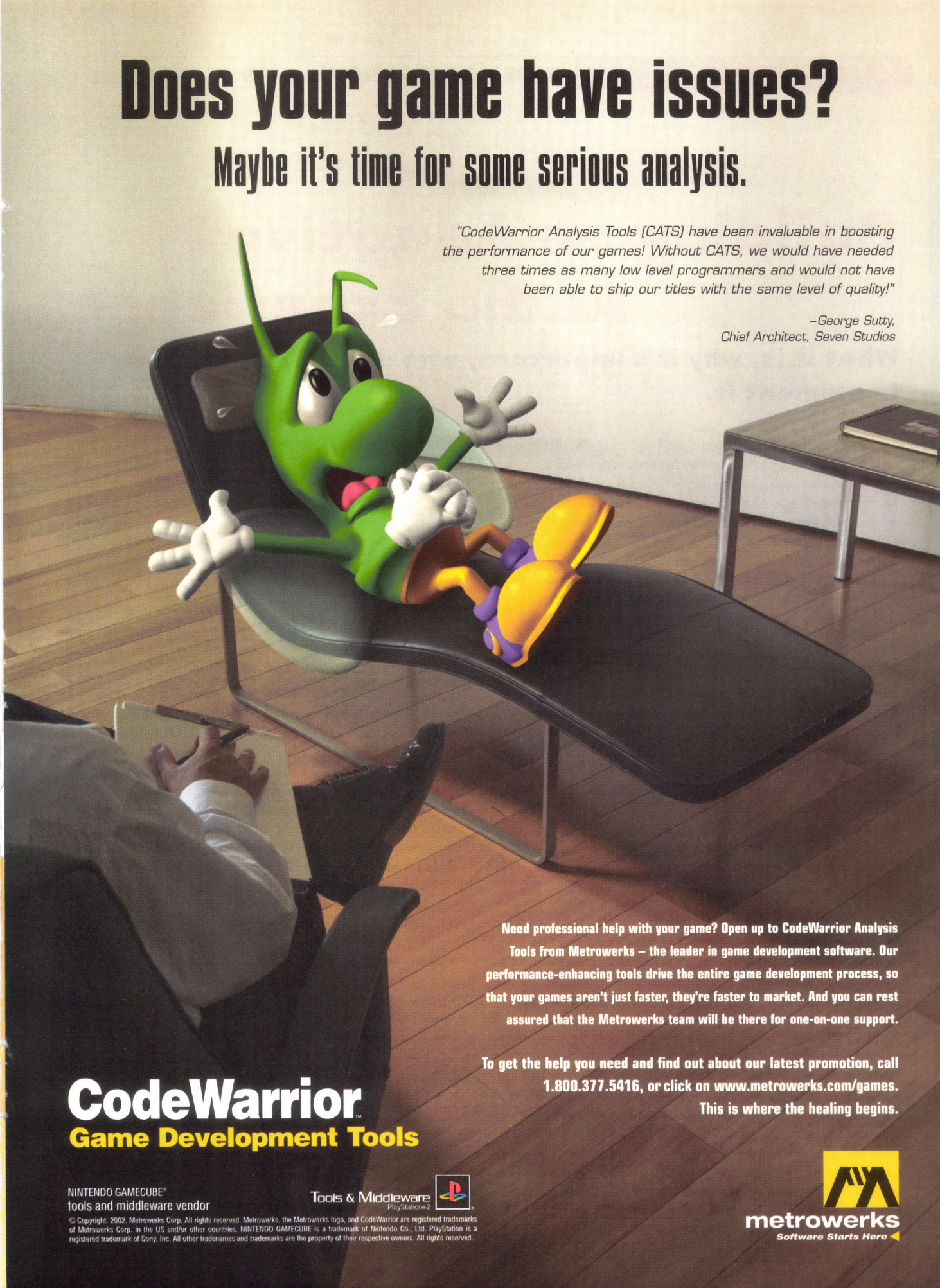
And finally, lest any fans of Charles Lutwidge Dodgson, better known as Lewis Carroll, protest, I admit that anyone who can work card games and chess into his books has some knowledge of games. But I stand behind what I said about Oscar Hammerstein II.



NOAH FALSTEIN | *Noah is a 22-year veteran of the game industry. You can find a list of his credits and other information at www.theinspiracy.com. If you're an experienced game designer interested in contributing to The 400 Project, please e-mail Noah at noah@theinspiracy.com (include your game design background) for more information about how to submit rules.*

Does your game have issues?

Maybe it's time for some serious analysis.



"CodeWarrior Analysis Tools (CATS) have been invaluable in boosting the performance of our games! Without CATS, we would have needed three times as many low level programmers and would not have been able to ship our titles with the same level of quality!"

–George Sutty,
Chief Architect, Seven Studios

Need professional help with your game? Open up to CodeWarrior Analysis Tools from Metrowerks – the leader in game development software. Our performance-enhancing tools drive the entire game development process, so that your games aren't just faster, they're faster to market. And you can rest assured that the Metrowerks team will be there for one-on-one support.

To get the help you need and find out about our latest promotion, call 1.800.377.5416, or click on www.metrowerks.com/games. This is where the healing begins.

CodeWarrior™
Game Development Tools

NINTENDO GAMECUBE®
tools and middleware vendor

Tools & Middleware
PlayStation®2

© Copyright. 2002. Metrowerks Corp. All rights reserved. Metrowerks, the Metrowerks logo, and CodeWarrior are registered trademarks of Metrowerks Corp. in the US and/or other countries. NINTENDO GAMECUBE is a trademark of Nintendo Co., Ltd. PlayStation is a registered trademark of Sony, Inc. All other tradenames and trademarks are the property of their respective owners. All rights reserved.

metrowerks
Software Starts Here

Building an Adaptive Audio Experience

What it is, why it's important, who does it, and how to implement it.

Just what is adaptive audio? Thomas Dolby Robertson said it best: "Adaptive audio systems provide a heightened user experience through a dynamic audio soundtrack which adapts to a variety of emotional and dramatic states resulting, perhaps, from choices the user makes."

The big surprise? He said that about five years ago. Funny for a bloke who goes from mainstream pop ("She Blinded Me with Science," in addition to his lesser-known but even more cool "Budapest by Blimp") to interactive audio engines (Beatnik) to mobile phones (the new Beatnik).

Want an even bigger surprise? No one has really documented adaptive audio (or AA, as some call it). Papers have been written skirting around it, conjecturing and theorizing, but no methodologies have been offered. It's time to change that.

In this article we'll be talking to a few old hands and a few new ones about their techniques and how they've worked, and hopefully they'll offer inspiration to sound and music people out there looking to add spice to their production efforts. We'll also talk about a group devoted to furthering adaptive audio and the man behind it.

What Games Use It, and Who Has Done It?

Potentially, all games can use AA. Let's see how by examining a title that uses a fairly straightforward design, RUSSIAN SQUARES, a puzzle game with a soundtrack by Guy Whitmore. You can see Guy's RUSSIAN SQUARES DirectMusic Producer project in Figure 1. The sequences are to the right (in piano-roll format with blue bars designating notes and note lengths), and how the sequences are used in the game is indicated by the list in the left column that contains "Motifs," "Patterns," and so on.

For Whitmore, this puzzle game was a wonderful opportunity to show how an adaptive score could work seamlessly with the core game design elements. The game starts with rows and columns of squares. The player's goal is to eliminate rows of squares by matching the color or shape of the squares. As the player eliminates rows, the difficulty increases. The time given decreases, and blockers are introduced to get in the player's way.

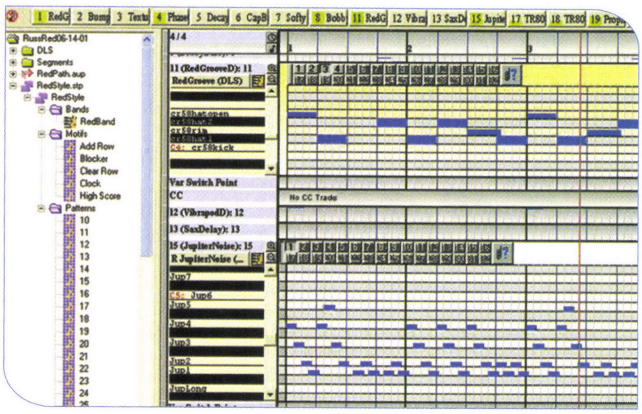


FIGURE 1. Guy Whitmore's RUSSIAN SQUARES DirectMusic Producer project shows both sequences as well as how they are used in the game.

The adaptive audio design. When creating a game score, the first step is deciding what the game calls for. Whitmore says he approaches each game without a particular adaptive technique in mind, but rather brainstorms in the abstract. That way the creative work leads the technical decisions, instead of vice versa.

Once he has an abstract idea of how the score could work, it's time to decide how to get the game engine to do it. Providing an adaptive audio design document as a complement to or even a part of the game design document is an important step in creating a highly adaptive score.

Adaptive elements. The physical core of RUSSIAN SQUARES' gameplay is the elimination and addition of rows, so Whitmore made this the core of the music functionality. As rows are added or taken away, the music responds by subtly changing: an instrument is added or subtracted, or the harmony or rhythm changes, for example. The music follows the overall pace of the player. To accomplish this response there

ALEXANDER BRANDON | *When he's not spending time with his wife, Jeanette, or gathering old game soundtracks for a massive compilation, Alexander keeps busy as the audio director for Ion Storm Austin. He is also a member of the board of directors for the Game Audio Network Guild (www.audiogang.org) and is on the steering committee of the Interactive Audio Special Interest Group (www.iasig.org). Contact him at alexb@ionstorm.com.*



Illustration by David Moore

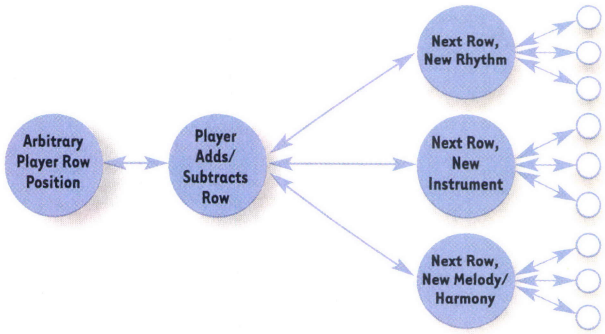


FIGURE 2. RUSSIAN SQUARES offers a perfect abstract representation of a simple nonlinear adaptive audio design.

GAME	GAME TYPE	TECHNIQUES USED
SHADOW OF THE BEAST 2	Side-scrolling action adventure	MOD-based soundtrack transitions using scripts
THE DIG	Graphic adventure	iMUSE soundtrack transitioning system between screens using flowchart-like interface
UNREAL/DEUS EX	First-person action/ first person RPG	MOD-based soundtrack transitions using scripts
REZ	Music-based action game	Soundtrack construction based on gameplay using MIDI transitions

TABLE 1. A selection of games from the early 1990s to the present and what adaptive audio techniques were employed by each one.

are about 70 music cells (DirectMusic calls them "groove levels") per composition. As the player completes rows, the music incrementally transitions to the next cell. Logical transition boundaries make those transitions musical and seamless. In this case they used the boundaries of the measures.

However, Guy found that simply moving along a linear curve from low intensity cells to high intensity cells didn't work aesthetically. In other words, adding layer after layer and increasing the tempo wasn't very musical. Given the game's style of music — modern electronica — it needed larger sections, instrumentation changes, rhythmic shifts, and tonal changes to keep it interesting. In this way the cells would progress in a multidimensional manner.

Figure 2 shows how the simple adaptive audio worked in RUSSIAN SQUARES to fit a fairly nonlinear design. As the player goes from row to row, the music can vary in a number of different ways using seamless transitions in DirectMusic. Each variation can also be used in addition to other variations for a much bigger listening palette of music, so all three of the variations shown in Figure 2 are possible in any order and combination as the player adds or subtracts rows.

The main method of keeping individual cells from getting monotonous was the use of instrument-level variation. Each cell is anywhere from two to eight measures in length and repeats as the player works on a given row. Within DirectMusic, each instrument can have up to 32 variations, which, when combined with other instruments, increases the amount of variation logarithmically. Most often, one to three instruments per cell use variation, giving the music an organic, spontaneous feel while preventing that looped feeling. Too much variation, however, can unglue the music's coherence.

With the examples in RUSSIAN SQUARES, we get an idea of

The Spectrum of Adaptability

An analogy I've become fond of using is that linear music is analogous to 2D prerendered art as adaptive music is analogous to 3D game-rendered art.

What did games gain from game-rendered art assets? The ability to view objects from any side or distance and the flexibility to create a truly interactive game environment, which put gamers in a more immersive and controllable environment.

Games gained the ability to view objects from any side or distance and the flexibility to create a truly interactive game environment, which puts gamers in a more immersive and controllable environment.

The analogy is very literal. Currently most game music is "prerendered": it's mixed in fairly large sections prior to being put in a game. Music that is more adaptive, on the other hand, is "game-rendered" — music components are assembled by the game as it is being played.

This brings up the concept of the spectrum of adaptability. On one end of the spectrum is linear, prerendered music, and on the other is music that is completely game-rendered. There are now many options for combining small, prerendered assets, such as WAV files, with assets that tend to be more flexible, such as MIDI files. Where on the spectrum a score lies depends on the game at hand, plus aesthetic decisions made by the composer and game designer.

Different degrees of adaptability are called for at different times within a game: a linear cutscene versus the ever-changing game environment. While I'm always experimenting with deeper and subtler levels of adaptability, I still use some prerendered assets. Why? Production values. All the adaptability in the world means nothing if the music doesn't sound good in the first place. I look for a balance between high production values and the flexibility of adaptive techniques.

— Guy Whitmore

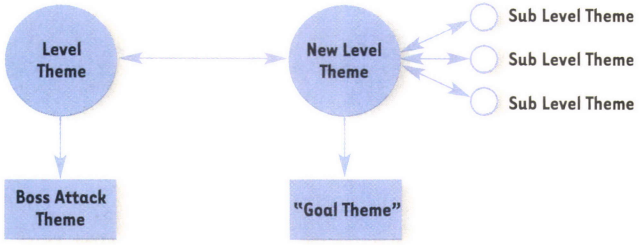


FIGURE 3. SHADOW OF THE BEAST 2 presents a more linear basic approach to AA, a step up from a single theme played in the background. In addition to the standard music transition upon changing levels, there were also sub levels activated by triggers and themes for boss creatures as well as goal-related themes.

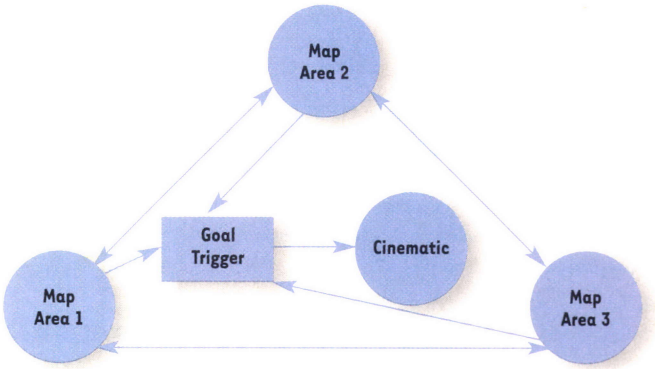


FIGURE 4. In THE DIG, the story is linear, but the game mechanics are not. The iMUSE system juggled different areas, triggers, and cinematic sequences as the player encountered them.

how to use AA in nearly any type of game using a similar system. But let's take a closer look at a few titles (listed in Table 1) with more complexity than RUSSIAN SQUARES.

SHADOW OF THE BEAST 2

Tim Wright, Jester Interactive's creative director, did the audio work for SHADOW OF THE BEAST 2 in 1990 well before most of us thought game audio was the cool place to be. Jester made SHADOW OF THE BEAST 2 for the Amiga for Psygnosis.

It was his first ever piece of commercial in-game music, and he was one of a two-man team. He composed the music and created the sound effects, and his brother Lee managed all the audio programming, compression, interrupt handling, and other details.

BEAST 2 featured basic adaptive audio in the sense that each game level consisted of sublevel areas. Each of these sublevels had a musical theme that would kick in when the player crossed certain triggers. In addition to themes for each area in the game, there were themes for surprise attacks from larger creatures, and also when certain switches were activated, to let players know they had done something significant.

Figure 3 shows SHADOW OF THE BEAST 2's approach to AA, which was more linear than RUSSIAN SQUARES' (shown in Figure 2). The music for SOTB2 was heavily based around each environment, which was fairly unusual for those days.

Ultimately, Tim won several Best Soundtrack awards for his efforts. In most of those cases, his adaptive soundtrack was specifically mentioned as one of the reasons for honoring the game's music.

THE DIG

THE DIG's sweeping score was used in conjunction with an engine that had used AA years before anything else like it happened on the PC. Michael Land wrote its music, and the now infamous iMUSE, or Interactive Music Streaming Engine, was used to blend his Wagnerian chords and melodies in streaming audio with adaptive techniques.

THE DIG, released in 1995, presented a fairly easy design with which to work. The player journeyed from screen to screen in the typical Sierra/LucasArts graphic adventure style and found ways of discovering and achieving goals to continue the story. Transitions for the soundtrack were based on moving from area to area, and various scripted events, such as accomplishing a goal.

There are two kinds of actions that represent these soundtrack transitions: immediate and conditional. The immediate

The "Adaptive Audio Now" Initiative

There is currently very little organized knowledge of adaptive game scores out there, which is why I decided to launch the Adaptive Audio Now (AAN) working group of the IA-SIG.

Interactive audio for games is still in its nascent stages of development. There is already a rich history of interactive game audio out there but very little information about its lineage and how it came to be. Many of us are creating our own rules as we go, simply because there are not yet standard techniques for creating an interactive game score. This working group and area of the IA-SIG web site fill a void in the current and ongoing state of adaptive audio in games and online media. It will follow the progress and evolution of adaptive game scoring as new games are released.

The approach is pragmatic in that published games and media will be its focus. The emphasis is on practical application, in the words of those who have created adaptive game scores. It will be a place to share and gather information, techniques, and production tips, as well as a place for others in the industry to find out what the interactive audio community is up to.

For more information about AAN, visit www.iasig.org/aan.

— *Guy Whitmore*

An Interview with Rez Creator Tetsuya Mizuguchi

Q. How long have you been involved in the videogame industry? What games have you created to date?

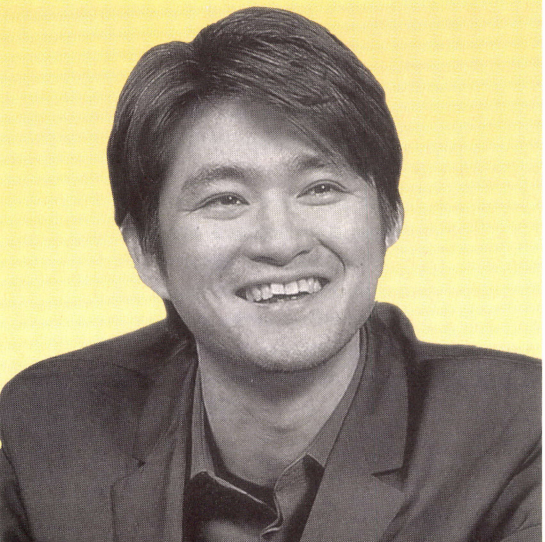
A. I have been working in the videogame industry for 13 years now. To date, I have been involved in the creation of the following games: Sega Rally (arcade), Sega Rally (Saturn version), ManxTT, Sega Touring Car Championship, Sega Rally 2 (arcade), Space Channel 5, Rez, and Space Channel 5, Part 2.

Q. Rez is a very unique and original game design. Was it difficult or easy to convince Sega to publish such a different kind of game?

A. Sega was at first perplexed by the concept of Rez; however, I would not say it was difficult to convince them. Sega is a company founded on innovation and creativity.

Q. In Rez, how did the audio work with the visuals? For example, how was the audio triggered and changed throughout the game?

A. In Rez, when you shoot down the enemy, notes of sound and visuals burst open. As you play, these actions compose an original score. If your gameplay is good, both the music and movie will become powerful, but if your gameplay is poor, the music will never climax. Try imagining that in a tube called a three-dimensional score, you are shooting objects that are MIDI packets filled with sounds and visuals.



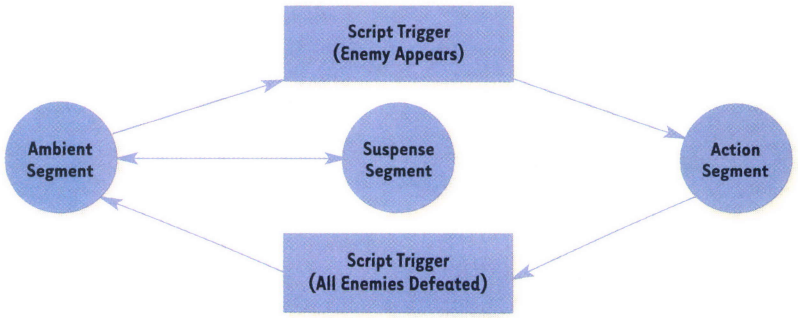


FIGURE 5. The AA design in Unreal used UnrealScript cues to transition between music tracks based on the action.

action allows the directing system to change the way the music is played (or change the piece of music being played) immediately — jumping to another place in the music, changing an instrument, changing volume, whatever.

The conditional action is what makes iMUSE so impressive when it's experienced. It allows the directing system to tell the sound driver to apply the change at the next musically appropriate time. This allows smooth transitions and change of instruments, as opposed to the sudden changes found even in today's games.

According to Land, iMUSE "started out as essentially a souped-up MIDI sequencer and has evolved into more a methodology than anything else." These days iMUSE is used only for streamed music, as a way to organize all the different music cues and see how they relate to each other by using a graphic layout. Land says the system allows the composer to specify musical responses such as transitions at a fairly detailed level.

Figure 4 shows how the iMUSE system straddled The Dig's linear story and nonlinear game mechanics, the standard model for graphic adventures since the King's Quest and Scott Adams Adventures days. The system provided seamless transitioning between areas, in any order. Players wandered between various areas consisting of a full screen each, where they solved puzzles and made choices to accomplish goals. As they walked to another area, the redbook soundtrack would cross-fade seamlessly based on measures and segments to ensure no harsh musical transitions. If the player accomplished a goal and a transition was needed to a cinematic quickly, the soundtrack would change using cross-fading.

Unreal/Deus Ex

Up until now we've only looked at 2D games. And, as with art and programming, 3D presents a much different challenge for AA. Now the composer and sound designers are presented with an environment that approaches reality from the human perspective, through emulation, simulation, or a combination of both. There are more axes of movement, more axes of vision, and more possibilities for game mechanics than ever presented in 2D. Some of the questions that the Unreal music team considered during the design phase included:

- Is music played in the background all the time or as a localized sound effect?
- If it is a localized sound effect, how is it represented? Simplified geometry within a level object, such as a building or landscape heightmap, or abstract geometry not related to the world and given its own parameters?
- Are transitions made with instant switching or various levels of cross-fading?

Because Looking Glass had done it successfully for the first time in a 3D environment in Ultima Underworld series, 1998's Unreal used adaptive audio using transitions similar to those in earlier games. Even with old design practices, careful considerations had to be made about when and where music was played.

Deus Ex followed suit in 2000 with a modified version of Unreal's engine. In these games the MOD format once again took hold, this time on the PC rather than the Amiga of Shadow of the Beast 2's day. Otherwise, games were rife

with the evil General MIDI format and all the tinny sounds it held. While the tunes were good, the sounds were not. MODs upped the ante and provided that much-needed step in between GM to streamed-audio quality.

UNREAL and DEUS EX used the Galaxy Music System to transition within a single MOD file from "ambient," "action," "suspense," and, for DEUS EX, "conversation" and "death" music tracks. Music used simple fades (alas, no cross-fades) within each level, depending on whether the player was wandering around, fighting, approaching a fight, and, in DEUS EX, talking to an NPC or getting killed. The system was about as straightforward as you can get, but it was a step up from everything else in the first-person 3D genre, with the exceptions of ULTIMA UNDERWORLD and FADE TO BLACK.

Figure 5 shows the AA for UNREAL using 1MB MOD files and scripted cues written in UnrealScript. Though simple, it sparked a few reactions, both negative and positive. Positive reactions were from people who thought the music added to the experience and made it feel more cinematic. Negative reactions came from people who thought the music took too much away from the realism. For instance, if a monster was near the player the game would call an action soundtrack to play, but in very rare cases the player wasn't even looking at the monster, so it acted as a cue not grounded in the game world, indicating danger. This proved to be a valuable lesson for later efforts.

REZ

Let's now enter a world where music and audio have a controlling share in a game's design and overall success. Over the last 10 years, games such as PARAPPA THE RAPPER, SAMBA DE AMIGO, FREQUENCY, and REZ have changed how game soundtracks are used and perceived dramatically. The groundbreaking REZ (2001) was the brainchild of Tetsuya Mizuguchi, head of Sega's United Game Artists studio.

Just what's the big deal about REZ? Not only does it look a lot like the movie *Tron*, which is cool in and of itself, but the game is a first-of-its-kind shooter where each shot contributes to the soundtrack, yet not one sound is out of place or interruptive in the way we've noted that certain AA transitions have been in first-person titles. REZ presents the player with a perfect AA environment, tailored to drive the gameplay itself. Simply put, REZ is a fantastic example of well-executed AA in action.

Looking to find out more about Mizuguchi's inspiration and design for the game, I asked a few questions of the man himself. See the Q&A sidebar on this page.

The Future of AA

As we've seen, adaptive audio has been around for a fair amount of time, considering the relative nascence of games. Still, because of the abstract nature of music, it remains one of the most difficult things to do in a game. For those of us writing music for puzzle, sports, or non-3D first-person games, the job is easier than in a more immersive first-person genre. But for all genres, if you want to go beyond these examples for your next titles, don't wait until you ship a game to try them. Drawing interaction diagrams and creating simulations is possible without writing one line of code.

It would be great if we were all multi-talented and could write our own 3D engine demos to see what really happens within a game engine when we apply adaptive audio techniques, but in the meantime, don't think that shipping a game will be the be-all end-all of adaptive audio research. Go for it with enough panache and you just might get the same attention as today's latest graphical bells and whistles. But gameplay remains the true judge. Base your audio design according to how the game plays as best you can. You and your players will be rewarded with a better overall game experience.

Q. Did you use a custom audio engine, or did you use licensed technology?

A. Both. For the creation of REZ, we used licensed technologies, such as ADX [Ed: ADX is a multi-stream sound technology made by CRI Middleware Co. Ltd.], and for other parts we developed original audio engines.

Q. How much influence did music composers have over the whole project?

A. In the case of REZ, our music composers had the same level of influence as the creators.

Q. Was there a lot of audio revision based on the design needs of the game, or did the musicians and sound designers understand enough at the beginning to make good judgments throughout the project?

A. At the preproduction stage, we spent every day making a lot of modifications. After the gameplay was finalized, we did not make as many adjustments. As we started to talk with the musicians after preproduction, I think everyone had a full understanding of the needs of the game design.

Q. What percentage of the project was spent developing and refining the audio engine? At what point in development did the audio engine become complete?

A. We spent approximately 50 percent of the time on the audio engine and 50 percent on developing the visual engine. The audio engine was completed at the stage when 70 percent of the visual development was already finished.

Q. How does music tell the player what to do in REZ, and do you think the same techniques can be used in other game types, such as adventure games?

A. Through the controller, the music in REZ communicates the pleasure of driving emotions and instincts to players. Of course, it is possible to translate this application to other games.

Q. Is there anything that you think could have been improved about the PS2 hardware?

A. In general, sound performance can be improved in consoles. If the hardware companies can improve the sound performance of game machines, they can also become audio-visual synthesizers controlled fully by MIDI. I believe that future game machines should always have a side like this.

— *A.B.*

Creating an Event-Driven Cinematic Camera: Part 1



Illustration by Greg Hargreaves

In the beginning, Auguste and Louis Lumiere made their first short movies without changing the position of the camera. It would take others, such as the magician Georges Melies, to stop and move the camera to create more dynamic scenes than could be done with a stationary, continuously running camera. Over the next century, cinematographers and editors have learned the best ways to film, cut, and transition between different shots to make the movie experience larger than life.

When it comes to camera setup, the dynamic nature of games puts us back to the days of Lumiere. With minimal or no control of the placement of the actors in a game, camera shots are often set as stationary or dragged behind one of the actors. Here and there, we see games that try innovative camera techniques — some that work and others that do not. One way to improve the chances of success is to take the film industry's century of experience and adapt it to our industry.

One method at our disposal in games is to create a system that automates camera placement and scene transitions (similar to the jobs of the cinematographer, director, and film editor on a major film production). In this, the first of a two-part series of articles on this subject, I'll look at automating the work the cinematographer does in setting up cameras to capture the best view of a scene. This will lay the groundwork for choosing among available scenes and camera shots required by the director and editor, which I will cover largely in part 2 next month.

Describing the Shot

Since the position and orientation of the actors is not known ahead of time, a method of describing the shot without exact positioning information is necessary. In order to arrive at a suitable set of parameters, let's look at the basic rules and descriptions that a cinematographer follows when setting up a camera shot. We are not searching for a complete description of how the shot is set up, but just for enough information from which to establish a position and orientation for the camera. For example, while we may eventually wish to implement different camera filters for interesting effects, what we care about now is mainly the field-of-view for the camera.

The most important rule of cinematography is called the 180-degree rule: The camera should not cross the line of action. The line of action is an imaginary line that partitions a scene into two distinct areas, usually going through the main actors or in the direction of movement. By not crossing the line between shots, a scene's screen direction and space is preserved. This idea can be extended to account for the three-dimensional nature of certain games by using a plane to partition space rather than just a line. This information is constant across all or several of the shots in a scene.

Once the line of action has been established, camera placement within the valid area follows what is known as the triangle system. Figure 1 shows the four basic camera positions for two actors: standard, over-the-shoulder, point of view, and profile. Each of these layouts has the same camera at the top point of the triangle, which is used for the establishing (or master) shot. The remaining cameras are meant to favor one actor over another. Figure 1 shows the approximate locations of the cameras on a two-dimensional plane, leaving the need for another parameter to describe the height or vertical angle of the camera.

Two remaining parameters involve the framing of the actor or actors on the screen. The first of these parameters is the shot size, which will determine the distance of the camera from the actor. Figure 2 shows the visible portions for a human actor in the common shot sizes. In addition to these shots, the long shot frames the actor with plenty of space between the edge of the screen and the actor. The other parameter determines the

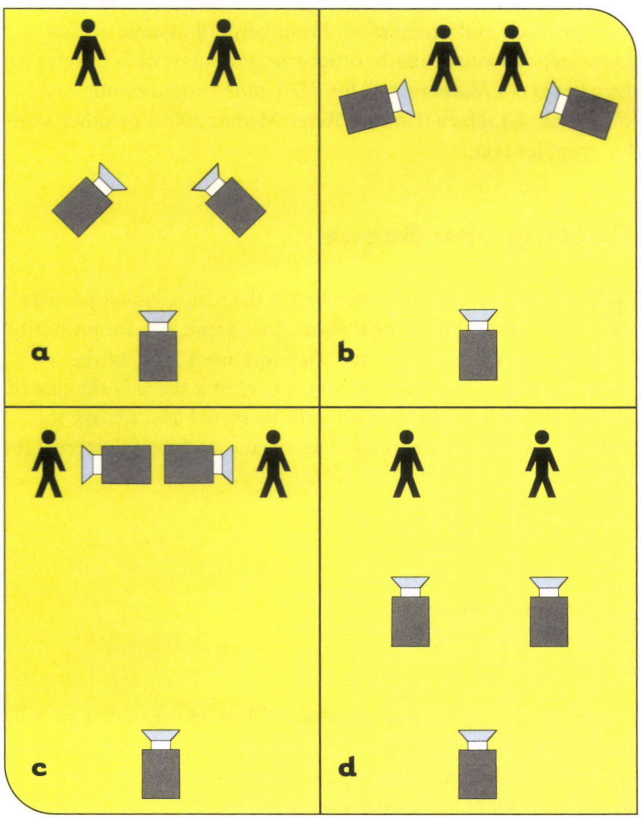


FIGURE 1. Common triangular camera arrangements: (a) standard, (b) over-the-shoulder, (c) point of view, and (d) profile.

BRIAN HAWKINS | *Brian began his career doing research at Justsystem Pittsburgh Research Center, where he focused on scripted character animation using natural language. He worked at Activision as the game core lead on* STAR TREK: ARMADA, *and contributed to* CIVILIZATION: CALL TO POWER *and* CALL TO POWER 2. *His last project was working for Seven Studios as lead programmer on* DEFENDER. *Brian holds a B.S. in mathematics and computer science from Carnegie Mellon University.*

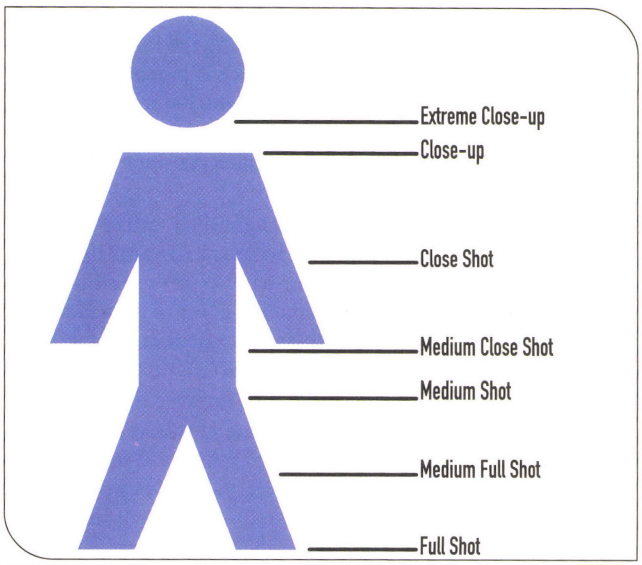


FIGURE 2. Visible portion of human actor for various shot sizes.

emphasis given to one actor or another, or the approximate screen space taken up by each actor. This information should be sufficient for us to move on to laying out the actual data structure and working out the equations for determining the final position and orientation. From here I'll assume a basic knowledge of vector math, otherwise, you may wish to refer to Eric Lengyel's *Mathematics for 3D Game Programming and Computer Graphics* (Charles River Media, 2002) or other similar graphics text.

Setting the Scene

The first data structure we need is the scene, which persists across several individual shots. In a scene, we are primarily concerned with the actors and therefore need a list of the scene's actors. The other important part of a scene is the line of action, discussed earlier, which will be stored as a matrix for reasons I will discuss shortly. This means a scene will store data similar to:

```
class scene
{
   // ...
   list<actor> actors;
   matrix line_of_action;
};
```

Let's take a closer look at the actor data and see what we will need to know about each actor. To simplify framing of the actor, we will consider only bounding spheres. However, we do need to use two spheres to model a human actor properly. The first sphere encompasses the entire actor, while the second surrounds only the head, or some other significant area for nonhuman actors. Finally, the orientation of the actor is important and should be included in the final data structure similar to this:

```
class actor
{
   // ...
   matrix orientation;
   sphere body;
   sphere head;
};
```

Now we turn to computing the line of action. As discussed earlier, we can consider this more a plane of action and therefore conveniently store it as a matrix. Although it's possible to store it more succinctly, the full matrix provides convenient directional vectors and is a standard data structure used in most games. The exact line of action depends on the number of actors in the scene; here I'll be considering either one or two actors. For one actor the line of action is the vector in the direction the actor is facing, and for two actors the line of action goes from the primary actor to the secondary actor. Use these as a right-vector for the matrix, and form the rest of the matrix using the average up-vector of the actors as a reference:

$$\mathbf{U} = \frac{\mathbf{U}_{primary} + \mathbf{U}_{secondary}}{2}$$

$$\mathbf{F} = \frac{\mathbf{R} \times \mathbf{U}}{\|\mathbf{R} \times \mathbf{U}\|}$$

$$\mathbf{U} = \mathbf{F} \times \mathbf{R}$$

$$\mathbf{M}_{CO} = \begin{bmatrix} F_x & U_x & R_x & 0 \\ F_y & U_y & R_y & 0 \\ F_z & U_z & R_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

For the position of the matrix, use the position of the actor for a single actor or for two actors the midpoint between them. Now that we have the data of the scene to shoot, we can move on to handling the elements of an individual shot.

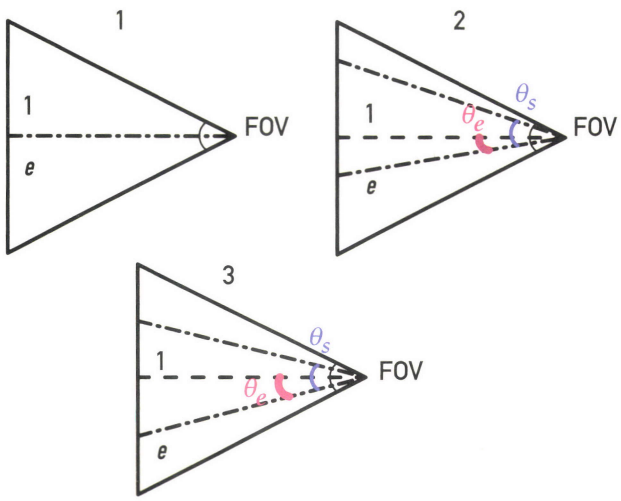
Emphasis

The first parameter we need to look at in the individual shot is the emphasis, because it will affect several of our following calculations. However, before we handle the emphasis we must choose what we want to emphasize. For our purposes, I'll assume that you can focus on the primary actor, secondary actor, or both. The focus is the center of the selected actor, which I will describe how to calculate when I talk about size. There are two ways we could handle focusing on both actors, the simpler of which is to use the midpoint between the centers. Later, I will discuss a slightly more complex approach that is particularly useful for over-the-shoulder camera shots.

The emphasis chosen specifies where on screen we wish to place the focus. Full-screen emphasis would place the focus in

EMPHASIS TYPE	OFF-CENTER ANGLE (θ_e)	SEPARATION ANGLE (θ_s)	UNIT PLANE LENGTH (e)
Full (see illustration 1)	0	0	1
Two-Thirds (see illustration 2)	1/6 FOV	1/2 FOV	sec(1/6 FOV)
One-Half (see illustration 3)	1/4 FOV	1/2 FOV	sec(1/4 FOV)

TABLE 1. Values for each emphasis type, where FOV is the horizontal field of view for the camera.



the center of the screen, two-thirds emphasis would place it at one-third of the way across the screen, and half emphasis would place it at one-quarter of the way across the screen. With that in mind, we need to convert the emphasis specification to values we can use in our calculations.

The three important values are the angle formed between the ray to the emphasis and the center ray, the angle between the emphasis of the main actor and the emphasis of the second actor (if any), and the distance from the camera to the intersection of the emphasis ray with the view frustum plane one unit from the camera. Table 1 shows the three main emphasis types and their respective values. With these, we are ready to move on to some actual calculations, starting with the camera distance from the focus based on desired size.

SHOT SIZE	HEAD/BODY INTERPOLATION (k_{hb})	SCREEN INTERPOLATION (k_s)
Extreme Close-up	0	3/2
Close-up	0	1
Close Shot	1/4	1
Medium Close Shot	1/3	1
Medium Shot	1/2	1
Medium Full Shot	3/4	1
Full Shot	1	1
Long Shot	1	1/2

TABLE 2. Interpolation values for shot sizes.

The first step to computing the distance from the focus is to determine the part of the actor to show. At the same time, we can compute the exact location of the focus. This location is the same as the center of the actor mentioned earlier. To do all this, we bring the head and body spheres into play by interpolating between them based on the desired shot size. Table 2 shows the interpolation percentages for several common shot sizes. Take these and plug them into the following equations:

$$\mathbf{c} = k_{hb}\mathbf{c}_b + (1 - k_{hb})\mathbf{c}_h$$
$$r = k_{hb}r_b + (1 - k_{hb})r_h$$

This gives the desired center, or focus, and radius values. Table 2 also gives the percentage of the screen we wish the actor to occupy, which equates to the ratio of the radius to half the screen height. Now we can form two similar right triangles, one using the radius as the far side and one using half the height of the frustum view plane one unit from the camera. Respectively, the adjacent sides are the distance we are seeking and the length value from Table 1. Since the triangles are similar, we know the following ratios are equal:

$$\frac{d}{r} = \frac{e}{k_s h}$$

Here, we can obtain h by taking the tangent of half the vertical field of view. Solving for the distance we get:

$$d = \frac{er}{k_s h}$$

This gives us the focal point and the distance the camera will be positioned from the focal point. Only two steps are left: the orientation of the camera location around the focus and the camera facing.

Angle

We start with the line-of-action orientation matrix, $\mathbf{M}_{CO}$, as our initial camera matrix before the angle changes are applied. Next, we need to determine the initial offset for the camera based on the distance we just computed:

$$\mathbf{v}_{offset} = \mathbf{M}_{CO} \times [0 \quad 0 \quad -d \quad 1]$$

With this as a starting place, we can break the remainder of the angle changes down into three separate rotations:

1. Rotate $\mathbf{M}_{CO}$ and $\mathbf{v}_{offset}$ around the up vector of $\mathbf{M}_{CO}$ by α.
2. Rotate $\mathbf{M}_{CO}$ and $\mathbf{v}_{offset}$ around the right vector of $\mathbf{M}_{CO}$ by β.
3. Rotate $\mathbf{M}_{CO}$ around the up vector of $\mathbf{M}_{CO}$ by θ.

SHOT TYPE	ANGLE
Standard	–45°
Over-the-Shoulder	75°
Point-of-View	–90°
Profile	0°

TABLE 3. Shot angles.

Now let's go over how to compute each of these angles. For the first pass, we start with a simple version of α used when we only care about ensuring that one of the actors is onscreen. The angle is chosen based on the shot type we want from those shown in Figure 1, although more shot types could be added. Table 3 shows the angles to use for each shot type.

The second angle, β, is the desired vertical angle from which to view the scene and should be between –90 and 90 degrees. That leaves us with only θ to determine, which is based on the emphasis angle, θ_e. The only modification is that we use $-\theta_e$ if we are looking at the primary actor and θ_e if we are looking at the secondary actor.

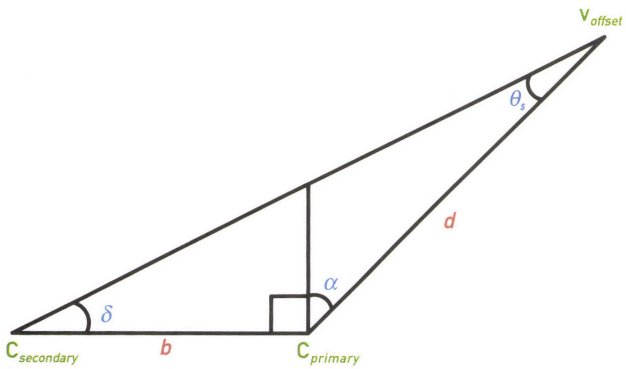


FIGURE 3. Camera layout for a shot framing both actors.

Finally, add $\mathbf{v}_{offset}$ to the focus location to get the final camera position. The orientation of the camera is $\mathbf{M}_{CO}$. That completes the basic shot. Let's look now at an alternative for shooting two actors that is more complex but produces a better shot.

Two Heads Are Better Than One

Sometimes you want both actors in the shot, for which the preceding methods didn't work. An improved method for framing in this case only need modify how α is calculated when working out the camera angles. Figure 3 is the desired result, and if we apply the law of sines to this, we get:

$$\frac{d}{\sin\delta} = \frac{b}{\sin\theta_s}$$

From this we derive:

$$\delta = \sin^{-1}\left(\frac{d\sin\theta_s}{b}\right)$$

Knowing that the angles in a triangle total 180 degrees, we get the following equation for α:

$$\alpha = 90° - \theta_s - \sin^{-1}\left(\frac{d\sin\theta_s}{b}\right)$$

Handling Obstructions

The main problem that arises once a camera shot has been calculated is an obstructed view of the actor or actors in the scene. There are a number of ways to handle this, of which I'll examine two of the most useful.

The first method is advantageous if the scene involves moving elements or a moving camera and you wish to ensure the actors are visible throughout. Start by rendering the actors first, storing the area of the screen and distance from the camera in which each actor is rendered. Now render the rest of the scene, skipping any objects that overlap one of the actors closer to the camera than that actor. Now go back and sort these actors from back to front and render them as translucent. This technique can be fairly expensive, especially for longer shots and complex scenes.

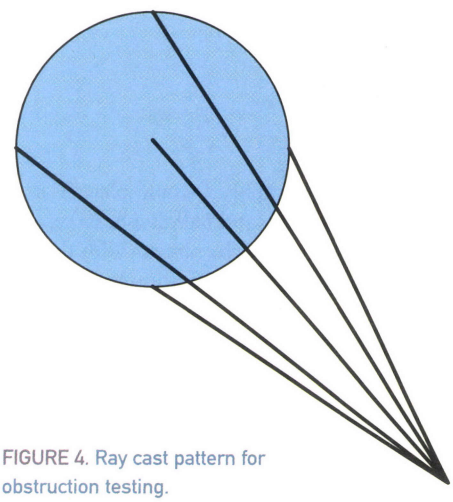


FIGURE 4. Ray cast pattern for obstruction testing.

There is a less expensive technique that we can employ during the setup of the shot. To use this method, you must determine if a shot is obstructed once it's chosen. A simple method is a line intersection test that uses a set of lines going from the camera to the bounding sphere of the actor, similar to those in Figure 4. Other patterns of lines can be used as long as the coverage is sufficient for larger objects. If there is an obstruction, an ordered set of shot alternatives are tested until one of the shots is unobstructed. This method allows smaller or moving objects to obstruct parts of the actors, but it is generally faster and less artificial than using translucency.

WE'RE MEMBERS OF THE IGDA BECAUSE...

We thought it was the International Girl Dating Agency. Then we found out that it was something even better...

To us, one of the IGDA's most important roles is recognizing the achievements of the unsung heroes of games development – the programmers, artists, designers, producers, etc – the people who sweat blood and tears to make great games. The Game Developers Choice Awards allow the developers themselves to be honored by fellow developers, who can appreciate how much effort has gone into the creation of a game and recognize new developments that will revolutionize the way the world plays games. It has been a great joy being involved with the Choice Awards over the past couple of years, and **we strongly believe in what the IGDA is doing to recognize and foster the art of game development.**

We always find it inspiring to meet so many creative people at our local IGDA chapter meetings, the Game Developers Conference, and other games conferences. Hearing the trials and tribulations of other developers makes us realize we are not alone in the eternal struggle to make great games!

The IGDA is on the side of the small fish in a sea of sharks.



THE KINGSLEY BROTHERS

CHRIS KINGSLEY, CTO
JASON KINGSLEY, CEO
REBELLION

The IGDA benefits the industry in so many ways. By encouraging a mature, constructive debate about the serious issues involved in creating video and computer games, without the distortion of tabloid hype or media frenzy. By encouraging game developers to communicate effectively with their peers in a friendly environment.

***All game developers should get involved with the IGDA,** because the IGDA is about individual game developers helping each other, and in so doing helping the entire industry. The IGDA makes it easy for all of us to get involved and give back to our growing community.*

www.igda.org/join

The IGDA is sponsored in part by:

Microsoft® Research Alias | wavefront®

igda international game developers association

On the Move

Movement is a very important part of games, and we would be lax if we did not consider it. First, we must consider how to handle moving actors. When we are focusing on only one actor, a simple solution is to follow that actor. If we cut to a different camera angle, we must obey the line of action at that time. However, it's not a problem if the camera rotates with the actor as long as the line of action is updated to reflect the new heading.

What happens when we are looking at two actors? There are several options, depending on the circumstances. If the actors are both known to be traveling in the same direction, the camera and the line of action can simply follow them as they did with only one actor. Another possibility, which could also be used with a single actor, is to let them travel out of frame before establishing a new camera shot and a new scene with a different line of action. This method allows the actors to move around in the shot as long as they do not stray too far. Cutting to a new scene is the only option if the actors travel in opposite directions, and the new scene will likely not have both actors in it.

One final form of movement that should be mentioned is camera movement. Up until now, we have treated shot changes as cuts, moving instantly from one camera position to the other. In some cases, a better or more interesting transition can be achieved by moving the camera smoothly from one position to the other. You could also use this to establish a new line of action within the same scene, but this should be used rarely.

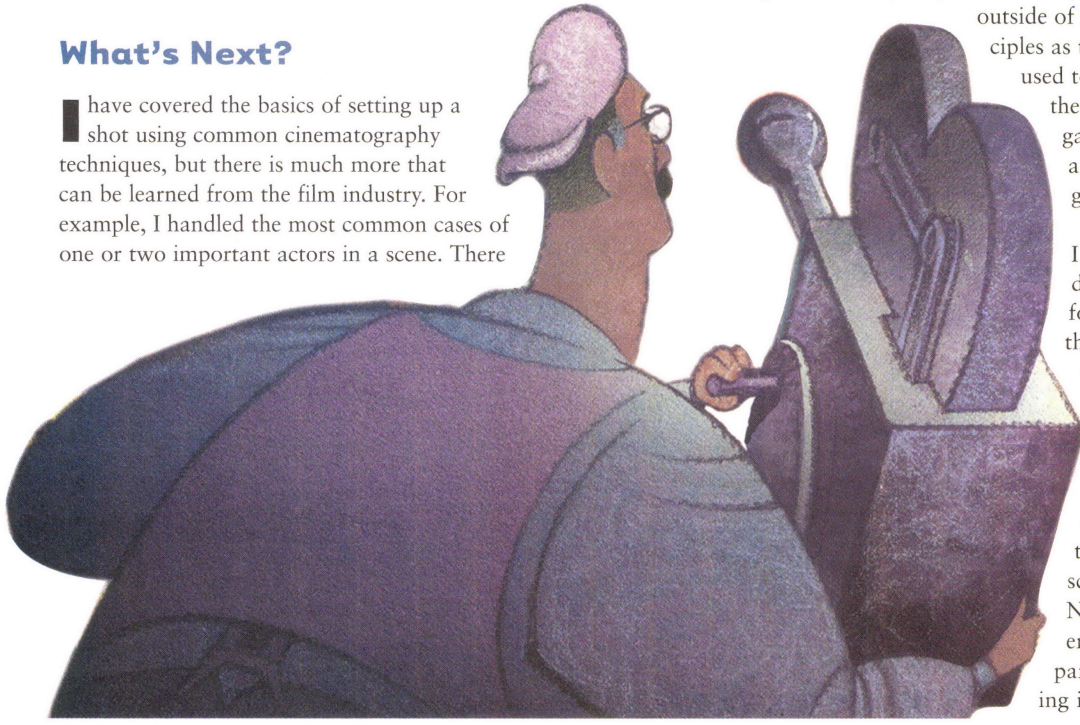
What's Next?

I have covered the basics of setting up a shot using common cinematography techniques, but there is much more that can be learned from the film industry. For example, I handled the most common cases of one or two important actors in a scene. There are occasions when three or more actors are important to a scene. See the For More Information section for additional resources on cinematography.

Another aspect that I have not discussed is the camera filter, which can add certain effects and moods to the scene. I dealt with one camera change, the field of view, because it was very important to position. However, it is also possible to perform color filtering, depth of field, and other more complex effects to add certain qualities to a shot. You should experiment with these effects to see if you can add that extra touch to your game.

A more complex issue that is rarely addressed in games currently is cinematic lighting. This is an extremely important part of filmmaking and can lend a professional touch to any scene. While lighting principles are well known in the film industry, in the game industry they are normally only applied to movies outside of the gameplay. Similar principles as those in this article could be used to help automate some of the lighting decisions for games, adding extra realism and production quality to a game.

However, the largest area I haven't touched on yet is deciding what shot to use for a given situation. With the addition of hints from programmers and level designers, it's possible to automate the process of shot selection. This involves deciding among available shots, picking transitions, and changing scenes when appropriate. Next month I'll take a closer look at this important part of our work on automating interactive cameras.



FOR MORE INFORMATION

BOOKS

Arijon, Daniel. *Grammar of the Film Language*. Los Angeles: Silman-James Press, 1976.

Katz, Steven D. *Film Directing Shot by Shot*. Studio City, Calif.: Michael Wiese Productions, 1991.

Lengyel, Eric. *Mathematics for 3D Game Programming & Computer Graphics*. Hingham, Mass.: Charles River Media, 2002.

GAME DEVELOPER

Lander, Jeff. "Lights ... Camera ... Let's Have Some Action Already!" Graphic Content, April 2000.



8 years of GameDeveloper for just $189

Each CD-ROM contains:

- ✓ Full text of feature articles, columns, and product reviews
- ✓ Articles, artwork, and diagrams exactly as they were printed
- ✓ All code and applications that accompanied the articles

Great content. Great prices!

NEW!

✓ May 2001 to April 2002	*$49*
✓ May 2000 to April 2001	*$39*
✓ May 1999 to April 2000	*$39*
✓ 1994 to April 1999	*$99*

Or get everything:

✓ Premier issue to April 2002	*$189*

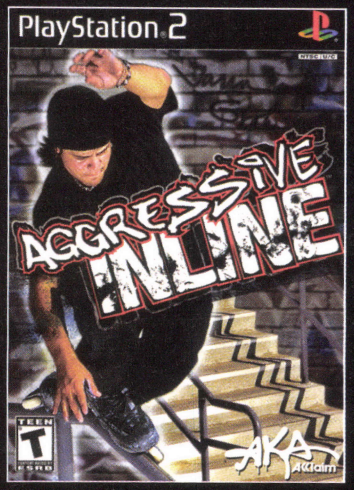
Order by phone:
800.444.4881 (US) or 785.841.1631 (international)

Order online:
www.gamasutra.com/store

Sponsored by: Maya™ Alias | wavefront

GamaNetwork

Z-Axis's Aggressive Inline



GAME DATA

PUBLISHER: Acclaim Entertainment
NUMBER OF FULL TIME DEVELOPERS: 25
NUMBER OF CONTRACTORS: 2
ESTIMATED BUDGET: Multi-millions
LENGTH OF DEVELOPMENT: 12 months plus 2 for Xbox, Gamecube
RELEASE DATE: May 2002
PLATFORM: Playstation 2, Xbox, Gamecube
DEVELOPMENT HARDWARE USED: T-10000 Sony dev tool, Sony Test debug stations, Xbox dev station, Gamecube dev station, 933MHz P3 PCs, Louisville Slugger XXL softball bat
DEVELOPMENT SOFTWARE USED: 3DS Max 4.2, Photoshop, 3D Paint, Bones Pro, Sony GCC, Visual Studio, SourceSafe, Metrowerks compiler, Premiere, assorted text editors
NOTABLE TECHNOLOGIES: Bink for Xbox video
PROJECT SIZE: 200 game code files, 350 engine code files, >10GB of art assets

Eight years ago, in his small Harvard dorm room, Dave Luntz founded Z-Axis. Soon after, the studio (or Dave, since he was the only employee) moved west to the San Francisco Bay area. From these humble beginnings, Z-Axis has gone on to ship nine titles, including John Madden Football '96, Thrasher Skate and Destroy, and Dave Mirra Freestyle BMX, on a variety of platforms to the U.S., Europe, and Japan. Currently the studio employs around 50 people constituting two development teams. I joined the studio more than five years ago as a lowly programmer and demoted my way to project manager.

Our most successful titles to date have been the Dave Mirra Freestyle BMX series, made in conjunction with publisher Acclaim Entertainment. In March 2001, Acclaim asked us to create an action sports game based on aggressive inline skating. At the time we had several projects on the table, but Acclaim offered a comfortable working relationship and a great business deal.

The inline game needed to be completed and blessed by Sony no later than May 2002, with Xbox and Gamecube versions finished a couple of months after that. All three platforms would also have PAL versions in multiple languages. That gave us a year to conceive and build a new type of game in the extreme sports genre. Twelve months is a tight schedule in which to produce a quality title, but luckily we had experience working within this time frame.

Our development team fell into three major areas of experience. The core art team had come over en masse from another developer. They had done some work on Mirra 2, so they knew the tool sets and export procedures, as well as the warts on the aforementioned systems. Lead designer Vince Castillo and I worked with this team to produce an in-house prototype. This experience served as something of an exhibition season for the team.

The programmers and most of the artists were wizened Z-Axis veterans. The remaining team members were game development rookies. Though many of them had worked in CG houses and had stared down brutal deadlines, they had no experience in the game industry. As a team, however, we were all hungry to make a quality game.

As we started initial design, we analyzed the heavyweights of the genre, Dave Mirra Freestyle BMX, a superior title with a winning record, and Tony Hawk's Pro Skater, the undefeated heavyweight that caused fear-induced loss of bladder control in lesser competition. We knew going into the project that only an extraordinary game could compete with these titles.

We took the things we really liked about Hawk and Mirra and then added things we really wanted to see in those games. For example, we observed on the Mirra products that when people played for the first time, they expected Hawk's game

RANDY CONDON | *Randy has been with Z-Axis for over five years, in the industry for seven, and ignoring his metallurgical degree for 14. He currently works as project manager with Z-Axis/Activision. Spam can be sent to randy@z-axis.com.*

control scheme. Plus, we liked the control scheme Hawk used, so we designed and incorporated a similar system.

Most of the additions from MIRRA were under-the-hood systems, including traffic systems, animation engines, some special effects code, the sound engine, the park-editor engine, and some other bric-a-brac. These hand-plucked goodies helped us tremendously in finishing the game within a year.

Next we concentrated on the new stuff. We wanted to go fast. Really fast. We wanted players to zoom around faster than they could within any other game in the genre.

We didn't want timed runs. We wanted to stay in the world, letting players do things at their own pace. When the urge struck, they could attempt the challenges. If players wanted to hang out and practice tricks, they could have at it.

As with many games in the RPG world, a player improved a skill by doing that skill well. For example, the more grinding a player did, the better his or her balance became. All seven player skills increased in this manner.

We also wanted players to alter the world. For example, completing certain challenges could unleash havoc. A player might grind across a rail where large fireworks rested. The fuses would ignite and send the explosives rocketing across the level. Exploding on an overpass, the rubble would subsequently settle into a giant bowl and set of ramps. We committed to having five to 10 of these large animations on every level.

The MIRRA BMX products featured a sophisticated level editor. Rather than leave this as an independent feature that players might discover, we wanted to incorporate it into the main game. We designed three of the levels to offer



challenges requiring the construction and then riding of a skate park. Finally, we wanted the game to have a strong sense of humor and fun.

We used motion capture to aid in animating the intricate tricks performed by inliners. Trying to figure out the grind variations these guys did on our own made us cross-eyed, so we motion-captured the best inline skaters in the world, who showed us how all the grinds and tricks worked. As a bonus, our designers and animator spent a week immersed with the athletes and culture of the skating community.

The motion capture took place at Camp Woodward, a mecca for professional extreme athletes and the adolescents who emulate them. Located in a remote part of Pennsylvania, the place has a million ramps, rails, and other devices inviting compound fractures. Campers ride all day on whatever they want, getting instruction from professionals.

So with the design complete and the motion capture sitting on hard disk, we jumped into production.

What Went Right

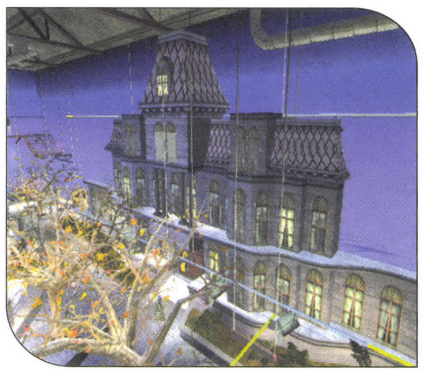
1. Make a plan, follow a plan. And remember to inhale and exhale, too. I'd omit this as obvious wisdom, except I keep hearing about projects that begin with a plan, follow a plan, make a new plan, ignore that plan while making a third plan, succumb to whimpering panic, and then jam whatever will fit onto a disk before the ship deadline.

We created a design and then made that design. I think we succeeded because the key members of the team contributed to the vision for the project. Lead artist Bill Spence, lead programmer Vince D'Amelio (a.k.a. Good Vince), lead designer Vince Castillo (a.k.a. Evil Vince), and I met early and often to formulate the design document. (The Vinces sat in the same cube, so we had to apply a virtue type to clarify our Vince identifications). In between sessions we would brainstorm on our own, then meet again. Bill relayed these sessions back to his art team, who produced concept sketches and color keys of important geometry, animations, and gameplay.









Concept art drawn in preproduction versus final art as seen in the Playstation 2 version.

The concept art wasn't just cubicle wallpaper. Level construction used the sketches as an initial blueprint. The final game remarkably resembles the concept sketches developed during the first months of the project.

As production progressed, we followed the credo "Gameplay is king," abandoning our initial credo, "The three-day work week is king." Our decisions were always examined in terms of improving gameplay. In the cauldron of production, gameplay can get overlooked, leading to wasted work or sterile levels. The artists and programmers were often the first to point out and fix their own problems. Because everyone understood how things should work, there was no destructive bickering or head bashing resulting from conflicting visions of the game. On occasions where debates arose, people deferred to designer Evil Vince, whom the team recognized and trusted for his talent for level design, despite his menacing first name.

2. Good producer, good publisher relations. Years ago, at a mostly forgotten GDC session, I remember a speaker insisting that your game was doomed without a good producer. Experience has taught me to endorse those words.

Shawn Rosen, slated to be AGGRESSIVE INLINE's producer, had the MIRRA BMX series under his belt, so we knew he was top-flight. This fact weighed heavily in our decision to choose this project.

In terms of third-party development with a publisher, what makes a good producer? Good producers are evangelical about the project within the game's publisher, pushing the project hard for marketing, advertising, and press interviews. The importance of these qualities cannot be overstated: The producer must believe in the developer.

They also put the project first, and are not egotistical or on a personal power trip. All their yelling and arguments are to make the product better, not testosterone-estrogen-induced posturing. On the other hand, the producer must not be a yes-per-

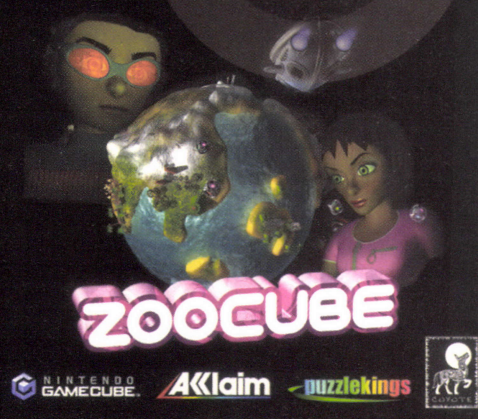
FREE YOURSELF TO DO THE CODE THAT MATTERS

Intrinsic Graphics builds its business on rock-solid technical support.

During the 30-day evaluation period, you can stress-test not only our product, but also our response time and our comprehensive knowledge base. When you purchase Alchemy, you receive the same excellent level of support that you have already tested, plus comprehensive technical training.

Advanced analysis and debugging tools within Alchemy allow lightning-fast response times at any stage in your game development process.

Intrinsic Graphics provides a forum, The Lair, available to the game development and art communities.
http://thelair.intrinsic.com/
Come by to visit.



INTRINSIC ALCHEMY®

THE GAME DEVELOPER'S ULTIMATE TOOL

sales@intrinsic.com

INTRINSIC GRAPHICS®

WWW.INTRINSIC.COM
650.966.6800

©2002, Intrinsic Graphics, Inc. All Rights Reserved. Intrinsic Alchemy and the Intrinsic Graphics logos are trademarks and/or registered trademarks of Intrinsic Graphics, Inc.

son to the developer. Agreeability can be pleasant during production, but a game must have objective feedback. Additionally, producers are indispensable when they can filter and focus the deluge of suggestions and comments from the publisher to the developer.

It is essential that a producer be dedicated to the project. Our game should not be one of many projects going on at once for the producer. Alternatively, the producer cannot be a slacker; he or she should be working as hard as we are.

Good producers provide creative feedback and offer good solutions to problems we encounter. They work well with the development team, not trying to be the designer, but offering good suggestions nonetheless.

Finally, good producers ideally have a track record of successful projects. They should be able to get the developer important demo dates, press meetings, stock meetings, and other details early, thereby minimizing the number of surprise builds and demos required by the developer.

Not all these qualities can be determined of a producer at the beginning of a project. The bottom line is to find a producer who believes in you as a developer, backs the game, and works well with the team.

We liked working with Acclaim because they let us do pretty much whatever we wanted. Because of the MIRRA successes, and because our milestones looked good and played well, they trusted us. They understood how we created quality product and that we thrived on minimum oversight.

3. Engine, tools, and animation system ready for use. We started with the graphics engine from MIRRA 2, which allowed us to prototype gameplay and export levels immediately. The animation system was in place and easily extensible to allow for our large level animations. For example, if we wanted a cruise missile to give a full-cavity search to a Macy's Thanksgiving Day Parade balloon, we just needed geometry and an appropriate animator. The technology was in place.

Our artists didn't have to wait for the engine or tools, so they had more time to create and fine-tune their levels. Likewise when motion capture data came in, the animator could quickly export it into our game engine. With no bike to complicate the data, motion clean-up was minimal compared to the BMX capture sessions.

Since the core engine was functioning, we were able to spend time optimizing performance for speed, texture management, and memory usage. Without these systems in place, we could not have produced a quality game within the schedule.

4. Profile speed, profile memory. Throughout the course of the project, we kept an eye on how much memory we were using and how well the game performed. We wanted performance at 60Hz at all times.

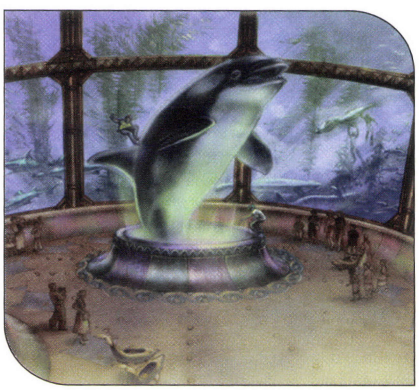
This isn't to say we constantly ran the game through a sophisticated profiler (although that did happen often, mostly near the end of production). When playing the game, there was an option to turn on the profiling bars to view performance. The bars extended horizontally across the top of the screen, with full-screen deflection indicating a full frame at 60Hz. The bars could show overall CPU, VU, or specific areas of code. The game code CPU usage, displayed as a green bar, was known as the "green bar of shame" by the engine programmers. The programmers could add or remove bars to display whatever code segments needed a look.

Thus we got a constant feel for how the game was performing. When something new got added, we'd know immediately if it was behaving as expected. The more technical level artists used the profiling to fine-tune geometry and even troubleshoot slow areas for optimization. It got to the point that when I watched or played the game, I didn't see an inliner moving around, all I saw were bars fluctuating on-screen.

Left to its own evil nature, memory will run out at the worst time, such as right before a milestone. As with performance, we constantly reviewed our memory usage. Listings were created detailing how much memory each module allocated. When something blew, we could compare listings from earlier builds and see what had — usually unexpectedly — grown. While this incurred a maintenance cost, it minimized the production-paralyzing episodes of unexpected memory crashes. We also found a few curious allocations that got eliminated.

The artists were assigned memory limits for level geometry, level textures, and animation sizes. They did a good job of staying within these parameters, minimizing memory crashes. On a tight schedule, we couldn't afford to waste days searching for memory savings.







These concept sketches were used to construct the models.

5. Locked physics early. Since our engine was in place, we could easily experiment with different gameplay possibilities. As preproduction ended, we established the inliner's speed, jumping, and grinding abilities. Having the physics in place defined the gap distances between quarter pipes for transfers (connecting jumps) and spacing between parallel rails so players could jump from grind to grind. These measurements provided the designers and level artists with a basic geometry palette. As soon as production began, they used variations on the basic themes to create intricate gameplay. The core physics did not change through the remainder of production.

Every level had a few places where a little extra oomph was needed to make a jump. The designers could tag any ramp or quarter pipe to allow different boosts when going airborne.

The significant tuning values were placed in menus available from the game. The programmers were free to work on greater problems while the designers fine-tuned parameters. When they got something locked down, a programmer easily made the change permanent. The tuning menus also allowed the producer to experiment with the game and give us quantitative feedback about his opinion on behaviors.

What Went Wrong

1. Design tasks were overwhelming. We designed an amazingly ambitious game. We thought we could do it all within our schedule; as happens with most games, we couldn't. We delayed cutting some features with the dream that they could still make it in. This false hope resulted in work continuing for several weeks and then getting cut. For example, we wanted a competition level in the game, with many flavors of events; coding and level construction continued for several weeks before reality dawned, and all that work had to be scrapped.

Our major delay was the time required to perfect level geometry. Since every surface and edge can be ridden, the design phase was quite sophisticated. Add to that the iterations required to perfect gameplay, and soon other tasks were suffering from lack of attention. Even with the advantages of a working graphics engine, this task took more time than any of us predicted.



Profiling performance of the Playstation 2 version of AGGRESSIVE INLINE. Full horizontal deflection indicates dropping to 30 fps. The red bar at the top is VU, the blue bar CPU, the green bar at the bottom is game code CPU usage. The red lines on geometry are edges tagged for grinding.

We were probably one level designer short of avoiding this bottleneck. The skill to visualize this type of gameplay is not common, and we had trouble finding additional help. The level artists did a phenomenal job assisting in this phase, taking the designs of Evil Vince and adding to his vision.

We didn't skimp on this area of the design, though, as we felt it was the strength of the game. We wanted players to ride the levels for hours and hours and still discover new lines and possibilities. Other features — the number of challenges, the amount of power-ups, and the removal of special gaps from the game — eventually got sacrificed for this gameplay.

Even with the cuts, each level featured between 20 and 35 challenges and a like number of power-ups. Testing a complete play-through was an involved process. As shipping approached, we spent numerous all-night sessions doing just that.

2. The challenge of challenges. In AGGRESSIVE INLINE, challenges are the tasks the player must perform to unlock new levels, enable secrets, and acquire other rewards. For example, to grind the hands of the clock tower, the player must first figure out how to get up to the clock face and then accurately perform the jump to the clock hands. We wanted lots and lots of challenges like this in the game, but we underestimated the work required to implement them fully. Every milestone involved marathon sessions to complete the challenges.

We used a proprietary scripting language to code our challenges, which was assigned to Dave Nelson, a junior programmer new to the game industry. Because of the multitude of tasks involving the other programmers, our newbie had to complete these tasks independently.

While Dave completed the job admirably, more oversight would have improved the process. Some of the script could have been off-loaded to code functions, while sage advice from experienced programmers could have eliminated redundancies and bugs in the scripts. By the time some of these improvements were made, we were past alpha.

The other major delay was that all the geometry, animation, NPC models, sound, icons, and support functions had to be in place for a full challenge implementation. Slogging around to all the people involved took significant effort and time. When the pieces were ready for scripting, we were only a few days out from a milestone.

This process could have been improved by lengthening the schedule to implement the challenges. The artists and animators should have completed their work before the programmer implemented the challenge. Achieving this process remained difficult, as constant attention had to be paid to staying on schedule. The longer schedule would have meant other features would have been cut or reduced.

Making sure everyone gets the supervision they require can be a tricky thing to accomplish. We want the leads to do some production work, as they are some of our most talented people. On the other hand, without certain key bits of advice or oversight, people can wander off into the weeds and have to redo or throw away work. By increasing our understanding of the tasks and the team, we're confident we can strike the proper balance in future projects.





Scenes from the Playstation 2 version of AGGRESSIVE INLINE.

3. Late start on technical side of front end. Although we got the art side of the front end prototyped and signed off, implementation was delayed. To add to our woes, the GUI tools had not been battle-tested, forcing us to debug as we used them. The programmer responsible for the front end had also coded the player animations, so he was constantly jumping back and forth between tasks trying to get everything done and working.

The conclusion from this was that we should have started sooner and either simplified the design or distributed the programming tasks a little more effectively.

4. Features not implemented 100 percent. Throughout production, too many features were not completely implemented, due primarily to schedule pressure or inexperience. While certain features might have met the intent of a milestone, they would require more work down the line. This disconnect left the game with holes throughout most of alpha and prevented the fine-tuning of some challenges until very late in the project.

As discussed earlier, the challenges were a problem. Usually they were simplified or not thoroughly play-tested. Other culprits included some acrobatic moves such as handstands and vaults that worked in many places, but not all. Wall rides proved nagging and inconsistent through early and mid-production. As such, we couldn't implement as many challenges using wall rides as we had hoped. This wasn't a result of laziness or sloppiness, just that the designated people had other tasks to perform. In other cases the feature incompletion was a rookie mistake, where individuals didn't realize that the last 10 percent of the job takes 50 percent of the time.

Ultimately some challenges had to be simplified or eliminated. Many of our rookies (and some of the veterans) now understand the importance of taking tasks as close to finished as possible. This follow-through has to happen for the game to get the fine-tuning that makes it an exceptional product. In the future, if need be, we will rework the schedule or cut lesser features in order to improve feature completion. We believe in making fewer features perfect rather than making more features halfway.

5. PAL was not our pal. As with most games, the team worked insane hours to get the product completed, which initially in our case was the NTSC version of the Playstation 2 game. We completed that version within the initial target date of the publisher, but it left the team burnt out. While we finished the Xbox and Gamecube versions to the same level of quality, the pace wasn't as quick.

Moving forward, we will commit more people to producing these other versions. We had dedicated engine programmers for the different platforms, but that proved insufficient. Game-side programmers will build the project for all platforms, artists will export levels for all platforms, and we will improve the level tools to make simultaneous exporting easier. Going through the process has greatly improved our abilities at near-simultaneous platform development.

The European versions also didn't go as smoothly as anticipated. I misjudged the state of our strings and sent them out for translation too early. After the initial translations, we had to make many changes that led to headaches and delays.

We also ignored some basics for strings, such as making sure the translated text would fit in the space available. We spent a lot of time reworking menus or translated strings to fix length problems. The scripting language also didn't have a smooth transition for translations, resulting in a hack solution that proved time-consuming to work around.

Vision to Reality

As AGGRESSIVE INLINE started coming together, we knew we were working on something good, but how good we didn't know. The game passed Sony just prior to E3, and the reviews from the show — and subsequently from the release — surprised us pleasantly. AGGRESSIVE INLINE consistently received reviews at 90 percent or better. We felt our efforts were rewarded.

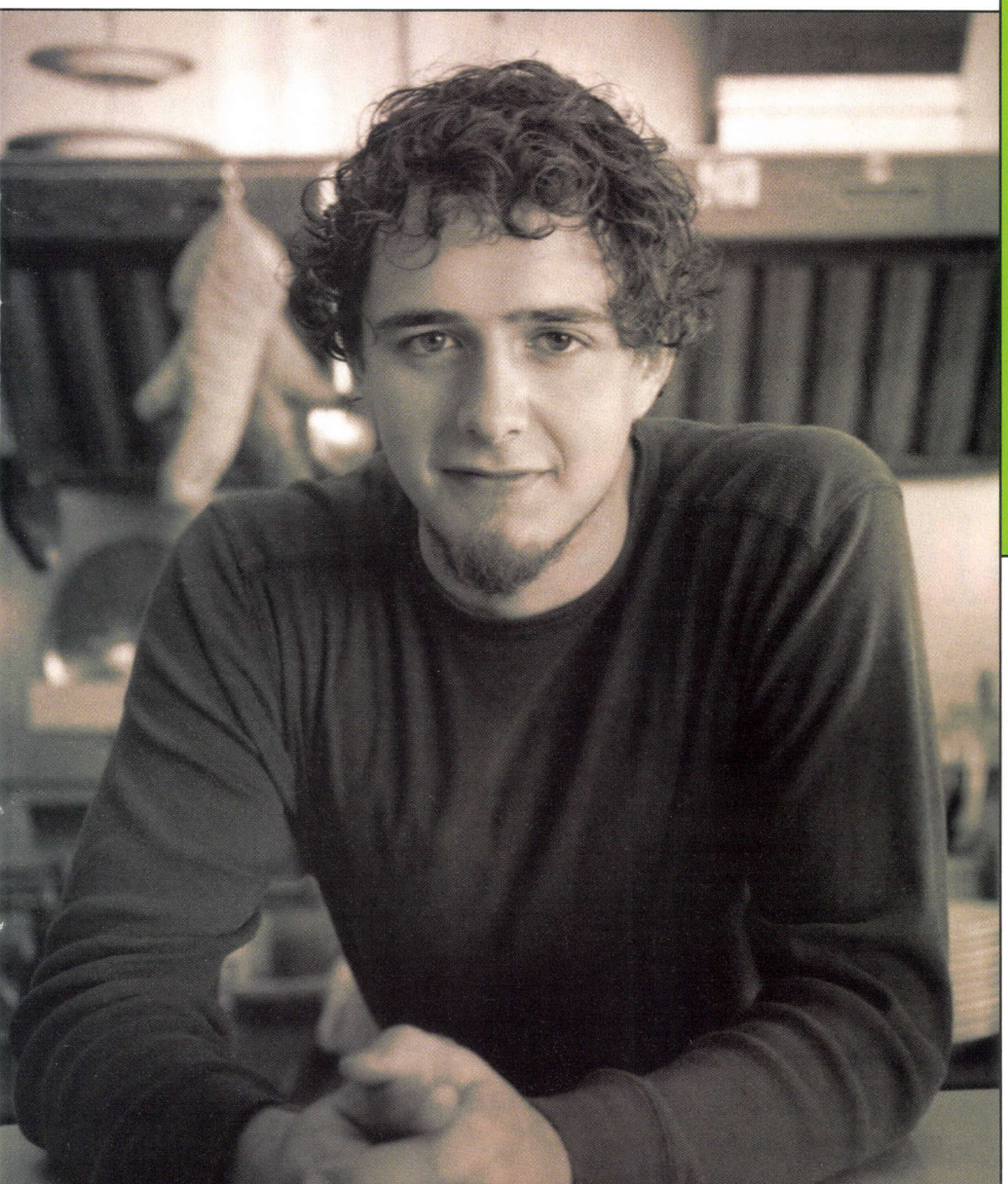
Looking back on the project, the thing I enjoyed most was brainstorming a vision and then transforming that vision into reality. We knew what we wanted to do and we did it. The team took the ideas and the game further than any individual could have done.

Our team remains intact. We have dissected the process of making this game and are committed to improving on the next. Can we keep it up? I believe so. We're a competitive bunch that likes to win. As long as we don't kill each other playing basketball or extreme Ping-Pong (don't ask), Z-Axis will continue making high-quality games and refining our development process.

Gamasutra.com

Gamasutra has always been the site where top game companies look for talent. Did you know you can post your resume on Gamasutra and let companies like Electronic Arts, Sony, LucasArts, Turbine, Relic and Nintendo come to you?

Make the next move in your career and post your resume on Gamasutra.
www.gamasutra.com/resumes



Don't play games with your career.

Why?

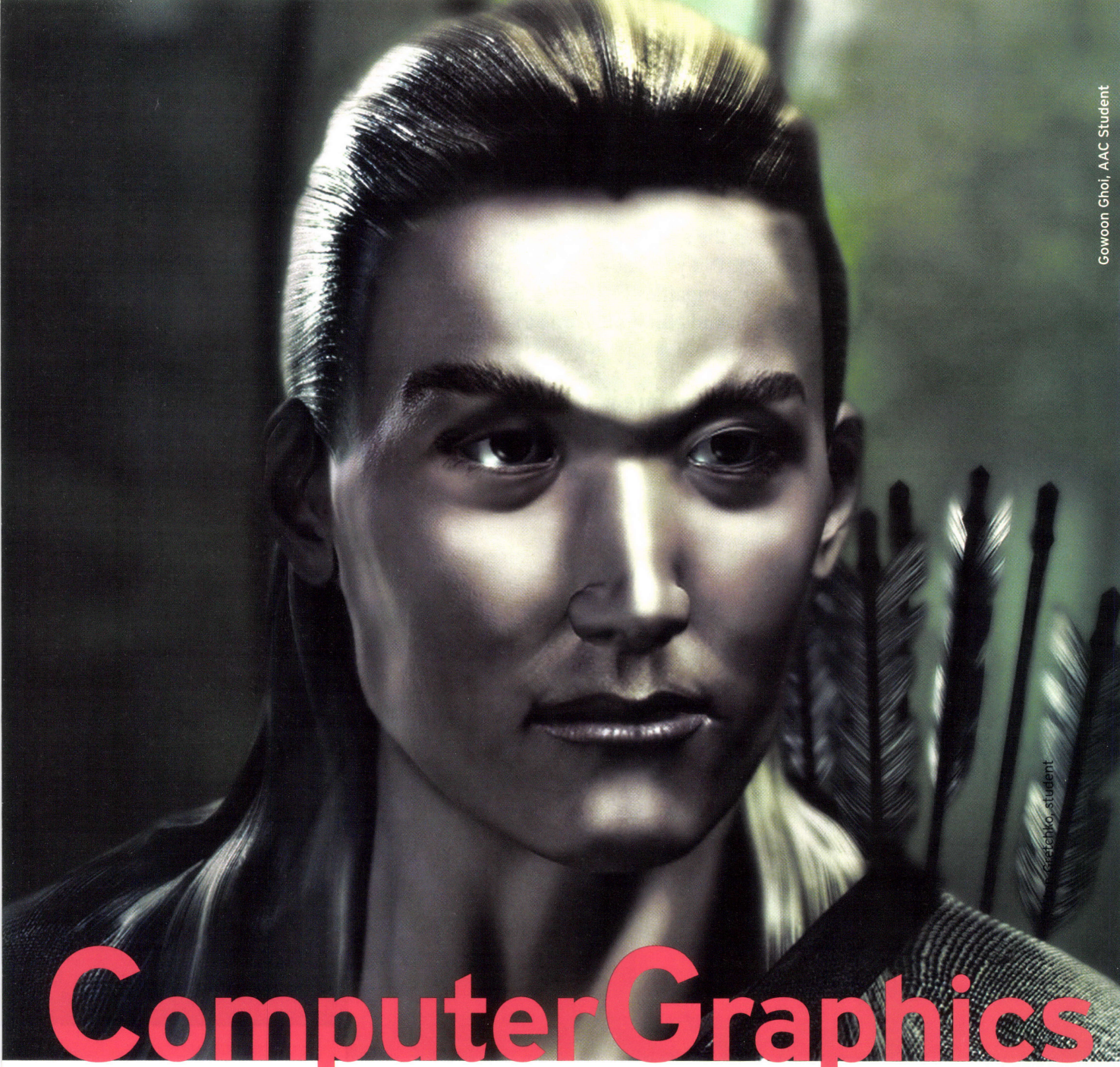
Focus. Our targeted community ensures that you'll only hear from game companies.

Flexibility. Want to remain anonymous? Don't want to hear from recruiters? No problem.

Control. Get results based on the technologies, platforms and tools you specify.

IT'S FREE.

GamaNetwork



ComputerGraphics

- 2D & 3D Animation
- 3D Modeling
- Character Design
- Computer Arts
- Computer Illustration
- Digital Imaging
- Digital Media
- Digital Photography
- Gaming
- Visual Effects
- Web Design
- And More...

AA | BFA | MFA Degrees | Portfolio Development | Online Classes

Apply Now for Spring, Summer & Fall Semesters
High School Scholarships & Teacher Grants Available

AcademyOfArtCollege
1.800.544.ARTS | www.academyart.edu

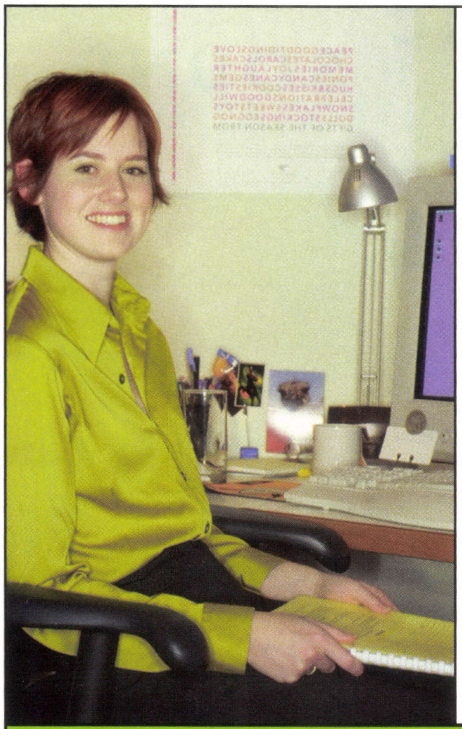
79 New Montgomery St. | San Francisco, CA 94105 | *Nationally Accredited by ACICS, NASAD & FIDER* | *Established in 1929*

CREATIVECAREERS

BLACK OPS® entertainment

Technical Director Needed

Black Ops seek a seasoned gaming veteran who will be responsible for the direction and day-to-day management of a project - team of programmers. Work in conjunction with the project's Art Lead(s), Designer(s) and Producer(s). Ability to adhere to demanding project schedules and successfully meet targeted milestones throughout the development lifecycle. Previous responsibilities should include extensive 3D, real-time programming, game play – game logic design, structured game architecting, development and implementation. 5-8 years programming - technical management experience required. Prior responsibility as a technical director and/or lead game programmer with 4-plus shipped next-gen console titles required. Advanced C programming skills, 3D math, graphics, AI and physics experience required. Current tools design and development a very nice plus. Expert debugging skills required. BS/CS, BS/EE or equivalent experience required. Please email resume, salary history and references to: rmines@blackops.com. No phone calls, please.



Gamasutra.com

It's easy:

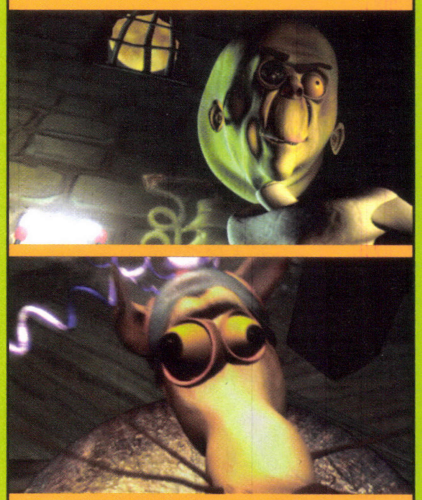
1. Just fill out a short form about your services and area of expertise, and you'll go up on the site the same business day.

2. **Basic listings are FREE**, and enhanced listings that let you upload logos, work samples and more are just a couple of dollars per day.

Game contractors: get listed.
www.gamasutra.com/getlisted/

GETEDUCATED

Savannah College of Art and Design



It's Alive!, Alias/Wavefront Maya,™ Adobe Photoshop,® Deep Paint 3D™ and Adobe After Effects®
Terry Ziegelman, St. Louis, Mo., second year M.F.A.;
Paul George, Valencia, Venezuela, second year M.F.A.;
Stephen Johnson, Charleston, S.C., B.F.A.;
Jamie Kirschenbaum, Reston, Va., B.F.A.

BACHELOR OF FINE ARTS
MASTER OF ARCHITECTURE
MASTER OF ARTS
MASTER OF FINE ARTS

Architectural History
Architecture
Art History
Computer Art
Fashion
Fibers
Furniture Design
Graphic Design
Historic Preservation
Illustration
Interior Design
Media and Performing Arts
Metals and Jewelry
Painting
Photography
Product Design
Sequential Art
Video/Film

AN INTERNATIONAL UNIVERSITY FOR THE ARTS
Savannah, Georgia
800.869.7223 | www.scad.edu

I said LISTEN.



BRICKMISTRESS
music and sound design

game audio never sounded this good.

www.brickmistress.com

"Love it, or else."



·MUSIC
·SPEECH ·SOUND
·AUDIO INTEGRATION

OMNI Interactive Audio

- Veteran team with over 10 years game audio experience each.
- Experience on **PlayStation 2**, **Xbox** and **Gamecube**, online **PC** games, mobile platforms.
- Complete audio package, or support for an overworked in-house audio team.
- Hassle free, efficient audio integration including audio tools.

www.OmniInteractiveAudio.com

we make models
we make textures
we make animations

we made art for these:

THE SIMS
THE SIMS - Livin' Large
THE SIMS - House Party
THE SIMS- Hot Date
THE SIMS - Family Vacation
Myth 3 - The Wolf Age
S3 - Unreal Tournament Edition
...and a lot more



we make art

NEW PENCIL

80 Liberty Ship Way, Suite 6
Sausalito, CA 94965
phone 415.339.1800
fax 415.339.1803
web www.newpencil.com

Your project is awesome.

Demand the best audio.



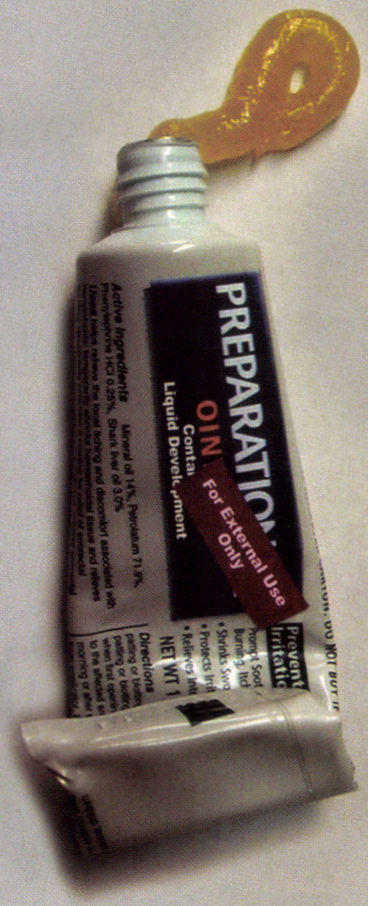
music • sound • voice

the sound department

415.454.8988

www.thesounddepartment.com

Embarassing Internal Problems?



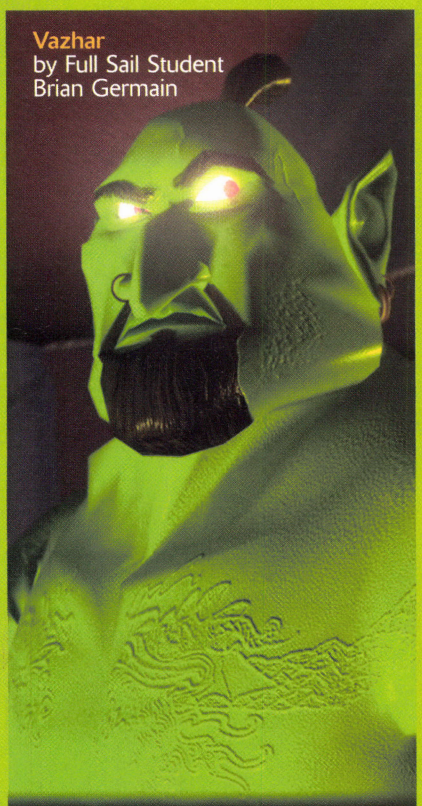
Get Fast Relief With Liquid Development.

LIQUID DEVELOPMENT

Liquid Development

The Only Art Resource You Need

www.liquiddevelopment.com



FULL SAIL
Real World Education

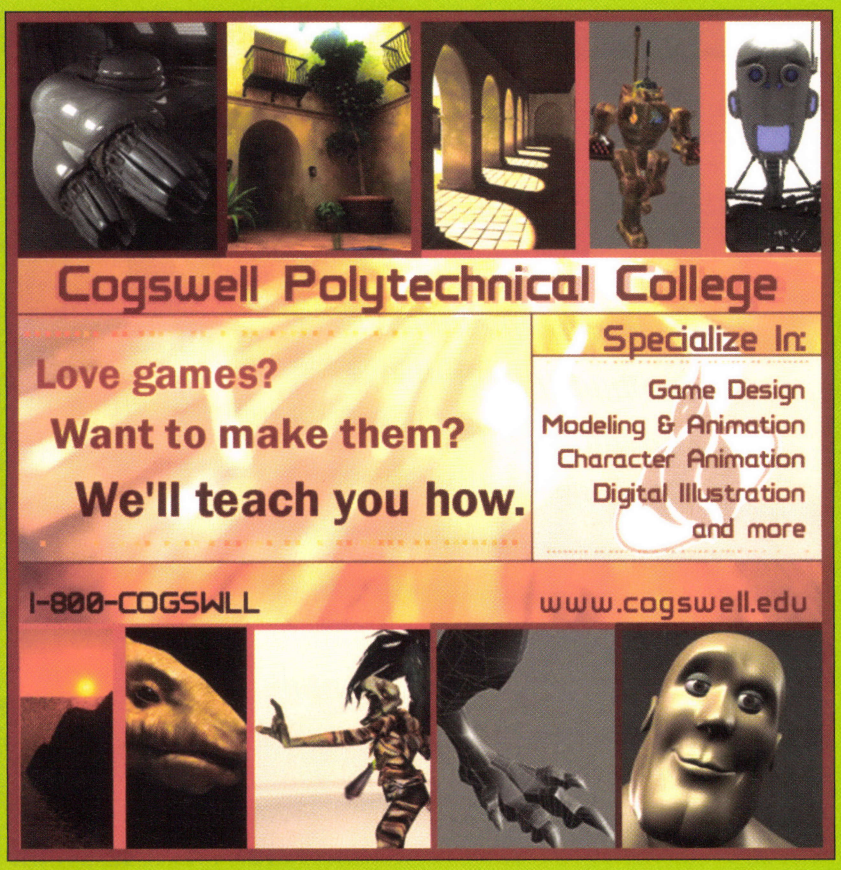
School of:
Game Design
Computer Animation
Digital Media
Film
Audio
Show Production

800.226.7625
www.fullsail.com

3300 University Boulevard
Winter Park, FL 32792

- Financial aid available to those who qualify.
- Job placement assistance.
- Accredited by ACCSCT.

©2001 Full Sail, Inc. All rights reserved.
The terms "Full Sail," "Full Sail Real World Education," and the Full Sail logo are either registered service marks or service marks of Full Sail, Inc.





continued from page 56

There is another reason why small teams are good. As team sizes increase, so team members spend more time communicating and less time developing. The more people are involved, the worse this problem gets. There are some notable exceptions, however. Austin Meyer is one. He produces X-PLANE single-handedly, and it is now the only general aviation flight sim to compete with Microsoft's. Geoff Crammond and his series of Formula One games is another.

It is possible to imagine not only new ecologies of creativity emerging, but also, perhaps, a whole new market for computer games. If enthusiastic and talented amateurs could produce an entertaining game, albeit without great graphics or sound, in six months, it could serve as a prototype for a more complete commercial game. It is even possible to see an intermediate market emerge somewhere between shareware and full retail, perhaps distributed over the Internet, with a much lower price than retail games. Games could be released into this market by well-known publishers who could supply a degree of quality assurance, testing, asset generation, and perhaps a small amount of financial support for the developer in advance of future sales.

The freeware and shareware development communities produce some amazing games. I was astounded at the quality of the entries for the Independent Games Festival this year. These developers' work is all the more astounding when you consider the difficulties they face in bringing their products to market. Imagine multiplying that enthusiasm by 10, translating it to the consoles, and supporting it with a vibrant commercial marketplace.

Bedroom programmers of the world, unite! You have nothing to lose but your Playstations.

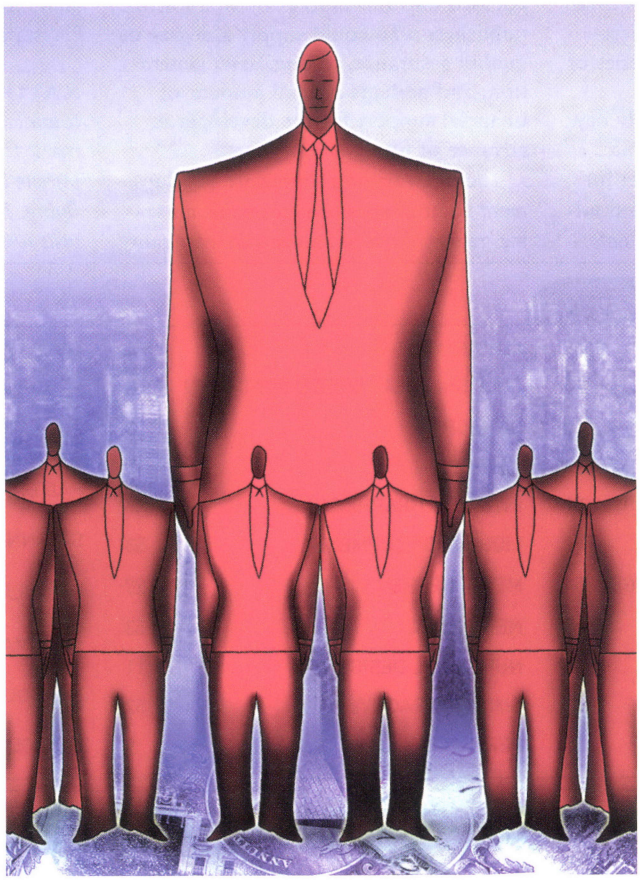
MATTHEW STIBBE | *Matthew founded IG, a London game studio, and ran it for 10 years before selling out to pursue his twin passions for writing and flying. E-mail matthew@stibbe.net or visit www.stibbe.net.*

ADVERTISER INDEX

COMPANY NAME	PAGE	COMPANY NAME	PAGE	COMPANY NAME	PAGE
3D LABS	13	FULL SAIL REAL WORLD EDUCATION	54	PROGRAMMERS PARADISE	7
ACADEMY OF ART COLLEGE	50	GAMESPY	23	QUALCOMM	C2
ANALOG DEVICES	17A, 25	IMMERSION	19	RAD GAME TOOLS	C4
BLACK OPS	51	INTRINSIC GRAPHICS	45	SAVANNAH COLL. OF ART AND DESIGN	51
BRICKMISTRESS	52	LIQUID DEVELOPMENT	53	SINGULARITY	52
COGSWELL POLYTECHNICAL COLLEGE	54	METROWERKS	27	SOFTIMAGE	3
COMPAQ	5	NEW PENCIL	53	THE SOUND DEPARTMENT	53
CONITEC	55	NUMERICAL DESIGN LTD.	11	VANCOUVER FILM SCHOOL	54
DICON 2002	C3	OMNI INTERACTIVE	52		
DTS	15	PERFORCE	9		



Games for the People by the People



In the old days it was quite common for a single person to create a computer game on his or her own. Some of the best-loved computer games were created in exactly that way. The first flight simulator was written by Bruce Artwick, SIMCITY was written by Will Wright, and the first million-selling computer game, ZORK, was written by two MIT students. The best of these games, games such as ELITE, ZORK, STAR RAIDERS, ROGUE, EMPIRE, or M.U.L.E. still stand up pretty well in comparison to today's offerings. The first game I designed, IMPERIUM, had one programmer and one artist, who did three SKUs in less than a year. Nowadays, almost all commercial games are produced by large teams.

There is nothing wrong with a large team. Sometimes they are absolutely necessary to get the games done to the standards we now expect in a time frame that is economically viable. When I was CEO of Intelligent Games, our smallest team was usually a dozen people. But in the two years since I left, it has struck me that the industry is losing out by sticking exclusively to this model.

I am not saying that we should all go back to the days of writing every game single-handedly. I like the games that I play now, and I wouldn't want to give them up. What I would like to see is a complementary and vibrant subculture of small games and small teams.

My manifesto: First, I would like to see every console come equipped with some kind of programming system. The generation before mine grew up with the Apple II, the TRS-80, and the Commodore Pet. My generation in England grew up with the BBC Micro and the Sinclair Spectrum. The next generation had Nintendo and Sega, which provided no chance to learn how to program (although today's PS2 Linux is a step in a good direction). Second, I'd like to see Microsoft offer a Visual Basic for Games and Sun create a games SDK for Java. Third, it would be exciting to see the open source movement do more to embrace games. Doing so not only would allow more large games to be created by teams of individuals working collaboratively, but also would allow beginners to learn from other people's code. My objectives are to increase the number of developers making games and to make the work easier in order to increase the number and variety of games in production.

The benefits of these simple steps are clear: the priesthood of game development would be thrown open to the masses — anyone could create games. This would mean new pools of talent from which to hire, the development of games for niches not served by today's commercial model, and, most importantly, an opportunity for the evolution of games to accelerate toward new game ideas and new genres. The last point is important, because almost all the game genres that we play today were first invented in the late 1970s or early 1980s. While technology and graphics have improved dramatically since that time, basic gameplay concepts have remained pretty static. This is because it takes so long to produce a game today that Darwinian natural selection simply doesn't have enough generations to do its work. We in the game industry often complain that our creativity is stifled, but it is hard to take risks on a multi-million dollar, two-year project.

continued on page 55

www.diconexpo.com

www.bcww.net

A New Platform for the Asia Pacific Market

International Contents Trade Show

DICON 2002 BCWW 2002

Explore the Korean Ultimate Game Industry with top decision makers from all over the world.
Korea has one of the world largest and the most advanced online game industries as a new emerging market.

Put yourself at the center of Asian
Digital & Broadcasting Content Market with boundless opportunities!

November 6-8, 2002 COEX Seoul, Korea

Organized by

KOCCA **Korea Culture & Contents Agency**

Arirang TV **The Korea International Broadcasting Foundation**

DICON 2002: Game, Animation, Music, Character, Comics, eBook and Mobile.
BCWW 2002: TV Program, Film, Drama, Documentary, Video, Satellite Service and New Media.
Followed by the successful Launch of BCWW 2001, DICON 2002 & BCWW 2002 will gurantee you with the most valuable marketplace for Digital & Media Contents.

DICON 2002 & BCWW 2002 Secretariat

KOCCA 5F, Hyundai Dream Tower, 923-14 Mok-dong, YangChun-gu, Seoul, Korea

Tel:+82-2-2166-2094 Fax:+82-2-2166-2097 www.diconexpo.com info@diconexpo.com

Chinese Representative
Contact: 劉成晴
Tel: +86-1360-977-8674
Fax: +86-20-8700-7359
E-mail: netching@hotmail.com

China Office (Beijing)
Contact: 權基永
Tel: +86-10-6518-7345 / 7346
Fax: +86-10-6518-7532
E-mail: quan86@hanmail.net

Japan Office (Tokyo)
Contact: Song-Hee Park
Tel: +81-3-5812-1537
Fax: +81-3-5812-1538
E-mail: shpark@kocca.or.kr

Japanese Representative
Contact: Sung Chil-Ryong
Tel: +81-3-3234-6121
Fax: +81-3-3234-6122
E-mail: chil@sannet.ne.jp

American Representative
Contact: Steven Nam
Tel: +1-212-736-2285
Fax: +1-212-736-2945
E-mail: bcwwusa@aol.com

The Best in Game Development Technology





Granny 3D 2.0

Granny 2.0 combines the **sophisticated dynamic 3D animation system** of its predecessor with a completely new and optimized run-time engine to deliver incredible performance in an extraordinarily small memory footprint perfect for consoles as well as PCs and Macs. Brand new, fully customizable and scriptable **MAX & Maya** plug-ins export all material and shader graph data, multiple texture and color channels for vertices, complete skinning information, and user-defined node attributes and scripts, all to a new, completely open, extensible, and easy-to-use file format.

With all this, plus utilities for advanced camera control, collision detection, animation and texture processing, MIP-map generation, and **Bink and S3TC** texture compression, Granny adds years worth of features to your 3D engine in just a few hours of integration time!





Bink Video Technology

The very best video codec - make your videos shine! Unrivaled quality (**better than DVD** and MPEG II). Up to **three times faster** than other true-color codecs. Perceptibly lossless, **8 to 1 audio compression**. Decompresses to DirectDraw, DIBSections, YUV overlays, or any other memory. Available for Win32, MacOS, Xbox, and now **Nintendo GameCube**. Think Bink!



Smacker Video Technology

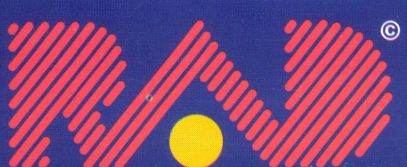
The best 256-color codec on the planet! Smacker is the codec of choice for **256-color games** (of course), games targeting Pentium 133s and below, games with video sprites or video transparency (**much faster in 256 colors**), or games with extremely high-resolution videos (800x600 and up). Smacker is available on Win32, Win16, DOS, and MacOS.





Miles Sound System

The ultimate sound system for PC and Mac! Miles supports 2D and 3D audio, MIDI with DLS, streaming, CD audio, **DSP filtering**, **MP3 playback (patent rights included)**, **Internet voice chat** (2900, 2400, and 1200 bps codecs), Creative **EAX 1, 2 & 3**, Aureal **A3D 1 & 2**, RSX 3D, DirectX 7, Dolby Surround, QSound, **super-fast software EAX emulation**, and more!



425 893 4300

www.radgametools.com

Powerful Technology. Easy To Use. No Royalties.